國 立 交 通 大 學

資訊學院

資訊科學與工程研究所

博 士 論 文

# 以 Microsoft Office 文件作資訊隱藏之新研究

# A Study on New Techniques for Data Hiding via Microsoft Office Documents

研 究 生: 劉 宗 原

指 導 教 授: 蔡 文 祥 博士

中華民國 九十九 年 七 月

# 以 Microsoft Office 文件作
# 資訊隱藏之新研究

# A Study on New Techniques for Data Hiding via Microsoft Office Documents

研　究　生：劉 宗 原　　　Student: Tsung-Yuan Liu

指 導 教 授：蔡 文 祥 博士　　　Advisor: Dr. Wen-Hsiang Tsai

國 立 交 通 大 學 資 訊 學 院
資 訊 科 學 與 工 程 研 究 所
博 士 論 文

A Dissertation Submitted to
Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy
in Computer and Information Science

July 2010
Hsinchu, Taiwan, 300
Republic of China

中華民國 九十九 年 七 月

# 以 Microsoft Office 文件作資訊隱藏之新研究

研究生：劉宗原　　　　　　　　　　　指導教授：　蔡文祥博士

國立交通大學資訊學院

資訊科學與工程研究所

## 摘　要

數位資訊處理與網際網路技術的快速發展，使資訊隱藏技術的發展愈為重要，其應用也更多元化。目前之研究偏重在影像、聲音、影片等檔案中藏入資訊，但在產官學界經常產生、使用、互通之Microsoft Office文件卻少有人研究探討。該類檔案之格式及特性迥異於影像、聲音、影片等檔案，需要嶄新之方法以達到版權保護、資料驗證及秘密傳輸等目的，極具研究價值。本論文針對Microsoft Office文件探討其特性並提出了六個在Office文件隱藏資訊之研究範圍，包括於Microsoft Office文件之文字中隱藏資訊、於文字編排中隱藏資訊、於嵌入之多媒體物件中隱藏資訊、於嵌入物件編排方式中隱藏資訊、於Microsoft Office文件輔助數據資料中隱藏資訊以及於實體檔案格式中隱藏資訊。本論文亦提出了六種具體的新的資訊隱藏方法及應用，可適用於常見之Microsoft Word、Microsoft Excel、Microsoft PowerPoint以及Microsoft Visio等檔案類型。

首先，本論文針對Microsoft Office 文件可多人編輯之特性提出在Microsoft Word文件中利用追蹤修訂資訊以及賀夫曼編碼(Huffman Coding)技術隱藏秘密之新方法。針對文件內容常被轉載之應用，我們提出多重適用性簽章方法(MUST)以及單樹根簽章方法(TRUST)兩種雜湊值及簽章處理方法並結合資訊隱藏技術以在Word文件中有效的達到轉貼資訊之來源驗證之目的，並提出二維多重適用性簽章方法(2D-MUST)及二維單樹根簽章方法(2D-TRUST)兩種二維雜湊值及簽章處理方法以在Microsoft Excel二維試算表文件中做轉貼表格之來源驗證，以及利用二維多重適用性簽章方法以偵測二維試算表文件內容可能遭竄改之應用。而針對文件內容常被剪貼、複製、收集之特性，本論文提出利用透明字元顏色及依權重統計偽隨機資訊隱藏順序之技術以在PowerPoint等文件中藏入隱密浮水印之新方法以達到來源追蹤等目的。另外，Microsoft Office 文件中常包含各

式影像、繪圖等，本論文提出了利用物件群組套疊關係以隱藏資訊以及利用創新的複合式一對一映射理論在影像中嵌入可逆式可視浮水印之新方法，而其中提出的可逆式可視浮水印方法可用於嵌入多種浮水印如單色不透明浮水印以及半透明全彩浮水印等。以上六種方法，皆為創新之作，實驗結果顯示論文提出的方法皆具有可行性及實用性。

# A Study on New Techniques for Data Hiding via Microsoft Office Documents

**Student: Tsung-Yuan Liu** **Advisor: Dr. Wen-Hsiang Tsai**

**Institute of Computer Science and Engineering**
**College of Computer Science**
**National Chiao Tung University**

## Abstract

With the advancement of digital information processing and Internet technologies, the field of data hiding has become more and more important, and their applications have become more and more diversified. Many techniques have been proposed for hiding data in images, videos, and audios, but there are relatively few researches devoted to data hiding in the popular Microsoft Office documents. Microsoft Office documents are in very different formats and have unique characteristics compared to images, videos, and audios, and so new techniques are needed for embedding data in such media for the purpose of copyright protection, covert communication, authentication, and so on. In this study, we investigate the characteristics of Microsoft Office documents pertaining to data hiding applications and identify six areas for researches of data hiding via such documents: data hiding via texts; data hiding via text formatting and layout; data hiding via multimedia contents; data hiding via multimedia formatting and layout; data hiding via auxiliary data; and data hiding via physical file formats. We also propose six specific new methods and applications for hiding data in Microsoft documents of Word, Excel, PowerPoint, and Visio.

First, exploiting the characteristic that documents can be written by multiple authors, a new method is proposed for embedding data in Microsoft Word documents for the purpose of covert communication by using change-tracking information and the Huffman coding technique. Then, to tackle the problem that contents in a document are often cited and included in another document and that there is a need to authenticate the fidelity and source of the cited content, a method is proposed in this study which combines data hiding techniques with two different hash value processing techniques – MUST and TRUST – that can efficiently verify the fidelity of cited contents in a Word document. Furthermore, two two-dimensional hash value processing techniques 2D-MUST and 2D-TRUST are proposed that allow quotations of the form of a two-dimensional table from a Microsoft Excel spreadsheet to be authenticated. Also, the 2D-MUST is demonstrated to allow effective fidelity authentication and modification detection of spreadsheet contents. To address the characteristic that contents within Microsoft Office documents are often moved, copied, and collected together, a new method is proposed for embedding invisible watermarks into slide presentations for the purpose of source tracking by using blank space coloring and weighted voting techniques. Finally, via rich media such as drawings and images contained in Microsoft Office documents, two data hiding methods are proposed, with the first using the different nested grouping relationships of objects to embed information in Microsoft Visio drawings, and the second method using a new generic approach of compound one-to-one mappings to embed completely-removable visible watermarks into images. The latter method was shown to be able to embed opaque monochrome watermarks as well as translucent full color visible watermarks, which is the first in publications to the best of the author's knowledge. Experimental results are included to demonstrate the feasibility of all the proposed methods.

# Acknowledgements

I would like to express my sincere appreciation to my advisor, Professor Wen-Hsiang Tsai, for his patience and kind guidance throughout the course of this dissertation study. I would also like to acknowledge the very helpful comments and suggestions from the members of the oral defense committee and also those from the reviewers for parts of this dissertation that were submitted for journal publication. Thanks are also extended to the colleagues in the Computer Vision Laboratory at National Chiao Tung University for their valuable help and comments during this study.

I would also like to acknowledge the financial support received from the National Science Council and the ZyXEL Scholarship, as well as the support from my current employer, Google, during the course of this dissertation study.

Finally, I am so grateful to my parents, brothers, wife, and children for their love, support, and endurance. This dissertation is dedicated to them.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1
# Introduction

## 1.1 Scope of Data Hiding Research

*Data hiding* is the study of embedding data into various media such that the information is accessible for later uses. The media into which data are embedded are called *cover media*[1], such as cover documents, cover images, and cover videos, and the resulting media with the data embedded are usually called *stego-media*[2], such as stego-documents, stego-images, and stego-videos. The data embedded into various media can be used for various applications, including covert communication, copyright protection, data association, media authentication, and so on [1]. The above-mentioned applications are described in the following.

1. Covert communication – data is hidden imperceptibly into a cover medium for the purpose of secret communication. For example, a sender may wish to transmit a secret message to a receiver secretly and so employs appropriate data hiding techniques to embed the message into a stego-medium. In this way, only the intended receiver can identify and retrieve the data hidden in the stego-medium. Data hiding for the purpose of covert communication is sometimes called *steganography*. Contrast to data protection techniques such as *encryption* that prevents observers from *knowing* the secret being transmitted, the goal of steganography is to *conceal the very act* of secret message transmission from outside observers.
2. Copyright protection – the data embedded into a cover medium is used to identify the copyright holder of the medium. The data embedded is usually called a watermark, and can be *visible* or *invisible*. A visible watermark advertises the copyright holder directly on a stego-medium, and can deter attempts of copyright infringements as a result, though at the cost of degrading the quality of the cover medium. On the other hand, a cover medium embedded with an invisible watermark is usually of a higher quality, but the copyright holder will need to prove the presence of the invisible watermark after the act of copyright infringement. In both the visible and invisible cases, the embedded watermark should be robust to removal attacks.

[1] Cover media can also be called host media or carrier media in the literature.

[2] Stego-media is often called watermarked media if the data embedding is for the copyright protection purpose.

3. Data association – the data embedded into a cover medium is the related information, such as the metadata, origin, and change history of the medium. Data hidden in this way facilitate the transmission and storage of the stego-medium along with the related information. It is desirable for the embedded data to be resilient against modifications to the stego-medium by programs unaware of the data association application.
4. Media authentication – the data embedded in a cover medium is used to verify the fidelity or the integrity of the stego-medium. This is important when a stego-medium needs to be transmitted over an insecure channel, where an attacker can alter the contents of the stego-medium.

A summary of the requirements for the above-mentioned data hiding applications is shown in Table 1.1 below. Is it noted that even if an attacker is not confident in the presence of hidden data in the covert communication application, a cautious attacker (often called an *active warden* in the literature) can nonetheless choose to perform small-scale transformations on all passing media. In this way, if there were no hidden information in the media, the transformations would be harmless to the communicating parties in normal circumstances, but if there were hidden information embedded, such transformations may be able to destroy the embedded data.

Table 1.1. A summary of requirements for different data hiding applications.

| *Data hiding application* | *Imperceptibility* | *Removal robustness* | |
|---|---|---|---|
| | | *Deliberate* | *Non-deliberate* |
| Covert communication | Y | * | |
| Copyright protection | | Y | Y |
| Data association | | | Y |
| Media authentication | | N | N |

In this study we investigate data hiding techniques for *office documents*, which are digital files used by the collection of programs in *office software suites* such as Microsoft Office and OpenOffice, with more emphasis on Microsoft Office documents. A number of office document types are in popular use today. The popularity of office documents is largely due to the versatile nature of the documents and the wide spread installment of office software suites. Some of the most common office document types are described in the following.

1. Word processing documents – these are very generic documents that can be used for all sorts of printable materials. It can be used, for example, in businesses for preparing letters, contracts, and reports, or in colleges for homework and publications. Such documents are most commonly processed using the applications "Microsoft Word" and "OpenOffice Writer", and are mostly commonly saved in the proprietary DOC format, the Office Open XML format (with the file extension ".docx"), or the OpenDocument text format (with the file extension ".odt"). Today's word processors allow text to be freely styled, and offer many productivity enhancing tools such as automatic detection or correction of typographical, spelling, or grammatical errors; automatic generation of tables of contents and lists of tables and figures; style management for format consistency; change tracking and commenting for multi-author collaboration; and so on.
2. Spreadsheet documents – these contain sheets of two-dimensional arrays of cells that simulate accounting worksheets. Such documents are most commonly used for numeric calculations or visualization of numerical values, and are mostly commonly processed using the applications "Microsoft Excel" (most commonly saved with the file extensions ".xls" or ".xlsx") and "OpenOffice Calc" (saved with file extension ".ods").
3. Slide presentation documents – these contain printable or projectable slides that can be used to aid a presentation. Such documents usually contain rich colors, animations, and various multimedia such as videos, images, drawings, or audios to emphasis the key points delivered via short sentences or phrases in the slides. The most commonly used presentation editing and viewing applications are "Microsoft PowerPoint" (which typically saves presentations with the file extensions ".ppt", ".pptx", or ".pps") and "OpenOffice Impress" (which typically saves presentations with the file extension ".odp").
4. Vector graphics documents – these are created using applications such as "Microsoft Visio" (most commonly saved with the file extension ".vsd") and "OpenOffice Draw" (saved with file extension ".odg"). Drawing applications typically supply *templates* containing various graphical objects as well as intelligent connectors connecting the

objects to allow efficient drawing of diagrams such as flowcharts and system architectures.

In the following sections, the motivation of this study is given in Section 1.2, followed by the contributions of this study in Section 1.3. Finally, the organization of this dissertation is described in Section 1.4.

## 1.2 Motivation of Study

Microsoft Office documents and other office documents are widely used in industry, government, and academia. Searching in Google reveals that over one hundred million of such documents are accessible online. Despite the popularity of office documents, currently there are few data hiding researches that address such documents compared to other media such as images, videos, and audios [2]-[6]. One reason is that earlier office documents are in proprietary binary formats. The only mention of data hiding in office documents known to the author that predates our study [7] did not attempt to understand the document format but instead just utilized the *slack space* at the end of the file or scanned in the binary file for consecutive bytes of 00's or FF's and replaced them with the intended payload for the purpose of covert communication.

Another reason why there is relatively little research in data hiding via office documents is that office documents can be extremely complex and can contain an assortment of heterogeneous, rich contents. The Office Open XML file format, for example, is a four-part ISO/IEC 29500:2008 standard [8] that contains 5560, 129, 40, and 1464 pages, respectively.

In this study, we focus on techniques that manipulate office documents at the *logical* level instead of modifying the underlying physical file formats. Such an approach is more generic and simpler, such that the proposed techniques can be applied to various office document formats. Also, data embedded into an office document in this way can often survive common editing operations as well as file format conversions. Details of accessing and manipulating office documents in the logical way can be found in Appendix A.

For data hiding to be effective, the characteristics of a cover medium must be taken into account so that the embedded data can be suitably blended with the cover medium for various desired purposes. For example, embedding data into an image would require different data hiding techniques than embedding data into an audio. Also, data hiding for different purposes has different considerations and hence requires different data hiding techniques. An invisible watermark embedded in a stego-image for the copyright protection purpose, for example, should be robust against common image operations such as resizing, cropping, and format conversion [9], [10]. On the other hand, data embedded for the purpose of media authentication do not need to be resilient against such modifications.

Office documents have several characteristics that are unique compared to other media such as images or videos, and require new techniques or approaches for effective data hiding. One characteristic of office documents is that they are frequently processed and manipulated by multiple parties. Examples of multi-party collaborations include workflows with office documents as attachments; collaborative authoring of journal manuscripts; and filling-in of forms in the office document formats. The collaborative nature of such use cases facilitates the application of steganography since it is natural for office documents to be transmitted between the collaborating parties. Data hiding applications such as data association and content authentication are also important considerations in such collaborative cases. In this dissertation study, we investigate data hiding techniques and their applications that take into consideration the collaborative nature of office documents.

Another important distinction between office documents and other cover media is the relative ease in copying, editing, and moving around parts of a document. This is especially evident in, for example, slide presentations, where the ordering of slides can be easily changed by drag-and-drop operations using a mouse. Also, it is common to compose a new set of slides by copying slides from several previously-authored slide presentations. The ease of reusing and manipulating portions of an office document poses challenges in copyright protection, data association, and media authentication applications, which are investigated in this study.

In summary, the goal of this research is to study the properties of office documents and propose new data hiding techniques that are suitable for office documents. Studies of data hiding via office documents are still few so far but are of great theoretical as well as practical importance because office documents are very popular and are created, transmitted, and consumed worldwide every day.

## 1.3 Contributions of This Study

In this study, we investigate and discuss the properties of office documents pertaining to data hiding applications and identify regions of office documents that may be used for data hiding applications. New techniques and approaches for hiding data via office documents are then proposed with applications that range from covert communication, authentication, data association, to copyright protection and for office document types that range from word-processing documents, spreadsheets, slide presentations, to drawings (a summary of the proposed methods can be found in Table 9.1 in the last chapter). In more details, the contributions of this study are as follows.

1. The properties of office documents pertaining to data hiding applications are investigated and six areas for researches of data hiding via office documents, including data hiding via texts; data hiding via text formatting and layout; data hiding via multimedia contents; data hiding via multimedia formatting and layout; data hiding via auxiliary data; and data hiding via physical file formats, are identified and discussed.
2. A new approach to covert communication is proposed by using change-tracking information, where the data embedding is disguised such that the stego-document appears to be the product of a collaborative writing effort. Text segments in a document are *degenerated*, mimicking to be the work of an author with inferior writing skill, with the secret message embedded in the choices of degenerations. The degenerations are then revised with the changes being tracked, making it appear as if a cautious author is correcting the mistakes. The change-tracking information contained in the stego-document allows the original cover, the degenerated document, and hence the secret message, to be recovered. It is proposed to use the Huffman coding technique for determining the choices of degeneration to make the method more innocuous, which is important for the application of covert communication. Also, one of the strengths of the proposed approach is that the extra change-tracking information added during message embedding is vital in a normal collaboration scenario, and so hinders ignorant removals by skeptics. Experimental results in Microsoft Word are presented to demonstrate the feasibility of the proposed method.
3. The problem of *quotation authentication* is investigated in this study to tackle the problem that contents in a document are often cited and included in another document and there is a need to authenticate the fidelity and source of the cited content. A new approach is proposed in this study that combines data hiding techniques with two different hash value processing techniques – the Multi-Use Signatures Technique (MUST) and the Tree-Root Uni-Signature Technique (TRUST) – that can efficiently verify the fidelity of cited contents in a document. Experimental results in Microsoft Word are presented to demonstrate the feasibility of the proposed method.
4. The problem of two-dimensional quotation authentication is described in this study. Such quotations can come from tables in a Word document or spreadsheets from Excel

documents. Two two-dimensional hash value processing techniques 2D-MUST and 2D-TRUST are proposed that allows efficient generation and verification of signatures required for authentication of two-dimensional quotations. Furthermore, it is shown that the 2D-MUST can be used for effective authentication and modification detection of spreadsheet content, and experimental results in Microsoft Excel are presented to demonstrate the feasibility of the proposed technique.

5. A new method is proposed for embedding invisible watermarks into office documents to address the characteristic that contents within office documents are often moved, copied, and collected together. In more detail, a watermark image is embedded imperceptibly into the slides of a slide presentation by partitioning the watermark into blocks and embedding them into the space characters existing in the slides in a repeating pseudo-random sequence. The embedding is achieved by changing the colors of the space characters into new ones which are results of encoding the contents and indices of the blocks. The embedded watermark is resilient against many common modifications on slides, including copying and pasting of slides; insertion, deletion and reordering of slides; slide design changes; and file format conversions. A security key is used during embedding and extraction of a watermark, such that if a presentation contains slides taken from presentations watermarked with different security keys, each watermark can be extracted reliably in turn with the respective key using a weighted voting technique also proposed in this study.
6. Data hiding in the drawings and images that can be embedded in office documents is also investigated in this study and two methods are proposed for drawings and images, respectively. The first proposed method embeds information into the structure of object groupings in a drawing. The objects in a drawing are grouped skillfully according to the data being embedded and the geometrical relationship between the objects. The groupings of objects in the stego-drawing are visually imperceptible, and the resulting stego-drawing is robust against translation, scaling, and rotation attacks. The proposed method can be used for data hiding applications such as drawing authentication and covert communication. Experiments conducted on Microsoft Visio drawings confirm the feasibility of the proposed method.
7. A new approach to lossless reversible visible watermarking in images is also proposed, in which deterministic one-to-one compound mappings of the pixel values in an image to those of a watermarked image is performed in such a way that the mappings tend to yield pixel values close to those of the desired visible watermark, making the resulting

visible watermark more distinctive. The compound mappings are proved reversible, to allow lossless recovery of the original image from the watermarked image. Different types of visible watermarks can be embedded as applications of the proposed generic approach, and two applications have been described where opaque monochrome watermarks as well as translucent color watermarks are embedded. Security protection measures by parameter and mapping randomizations have also been proposed to deter attackers from illicit image recoveries. Experimental results proving the effectiveness of the proposed approach as well as the invariability of the method when the images are embedded into Microsft Word or PowerPoint documents are included.

## 1.4 Dissertation Organization

In the remainder of this dissertation, the six regions for data hiding in office documents are explored in Chapter 2, along with surveys of related studies. In Chapter 3, the proposed method for data hiding via change-tracking information and Huffman coding is described. In Chapter 4, the new proposed approach to text quotation authentication is described, while the two-dimensional case is presented in Chapter 5. In Chapter 6, the proposed method for embedding invisible watermarks in slides of a presentation is described. In Chapter 7, the proposed method for hiding data in the structure of drawing object groupings is described. In Chapter 8, the proposed new approach for embedding removable visible watermarks into images is described. Finally, in the last chapter, conclusions of this study and some suggestions for future research are included.

# Chapter 2
# Six Areas for Researches of Data Hiding via Office Documents and Surveys of Related Studies

Office documents are very versatile and contain many types of contents, including rich text, images, drawings, videos, or even other office documents. We describe below six research directions using office documents for data hiding applications, and describe related works that can be used for hiding in the different regions.

## 2.1 Data Hiding via Texts

Most office documents contain texts, and so data hiding techniques such as *linguistic steganography* [11] that apply to the text itself can be used for data hiding via office documents. One approach to data hiding via texts is to generate the text content directly based on the data to be embedded, which is sometimes called *text mimicking*. By storing the text generated in such a way in an office document, the document can be used for the covert communication purpose because the intended receiver can easily extract the text contained within the office document and decode the text to extract the secret message contained therein.

A number of methods have been proposed in the past for text mimicking, such as using probabilistic context-free grammars [12], [13] for generating grammatically correct (though sometimes illogical) texts; or using several predefined *sentence structures* with swappable verbs, adverbs, adjectives, and other parts of speech [14]-[15] for embedding information. Figure 2.1 shows an example of a message generated by spammimic [16], a web-based steganography tool that uses context-free grammars to generate spam-like texts, with the secret message “Hello NCTU!” embedded.

Another approach to data hiding via texts is to apply semantically equivalent transformations of the text based on the embedded message. Examples include replacing words with their synonyms [15], [17]; performing *syntactic transformations* [18] like passivization (rendering a sentence into the passive form) and clefting (changing a simple sentence into a complex sentence with a main clause and a dependent clause)[3] [19] on a sentence's structure with little effect on its meaning; or performing one-way or multi-way[4] machine translations on a text [20]-[21].

```
Dear Friend , Especially for you - this breath-taking
news . If you are not interested in our publications
and wish to be removed from our lists, simply do NOT
respond and ignore this mail . This mail is being sent
in compliance with Senate bill 1916 ; Title 7 ; Section
302 . THIS IS NOT MULTI-LEVEL MARKETING ! Why work
for somebody else when you can become rich as few as
33 days ! Have you ever noticed society seems to be
moving faster and faster & nobody is getting any younger
! Well, now is your chance to capitalize on this !
WE will help YOU increase customer response by 160%
plus process your orders within seconds ! You can begin
at absolutely no cost to you ! But don't believe us
! Ms Jones who resides in New Hampshire tried us and
says "My only problem now is where to park all my cars"
! We are licensed to operate in all states ! We beseech
you - act now . Sign up a friend and your friend will
be rich too ! Thank-you for your serious consideration
of our offer !
```

Figure 2.1. A spam-like message generated with spammimic [16] with the secret message "Hello NCTU!" embedded.

A third approach to data hiding via texts is to use invisible characters or make small-scale modifications to the text so that the change is not noticeable. Examples of this approach include adding or removing spaces before or after punctuations and symbols; using two spaces instead of one and vice versa; introducing occasional typos or misspellings [22]; inserting non-visible special characters such as unused ASCII codes [23]-[24], directional formatting codes, or Unicode joiner characters [25]; or replacing characters by identically or similarly looking alternatives such as replacing a space by its *non-breaking* version [26] or using alternative character sequences that produce identical rendering [27].

As mentioned previously, it is possible to apply the aforementioned techniques for data hiding via office documents. The feasibility of this approach is demonstrated in Chapter 3,

[3] An example of passivization is to change the sentence "Renee gave a speech" into "A speech was given by Renee," and an example of clefting is to change the sentence "We are looking for Biwi" into "It is Biwi whom we are looking for."

[4] For example, translating a text from English to Chinese and then back to English, or from English to Japanese, to Korean, and then to French.

where the text in a Microsoft Word document is modified in a certain way using some of the techniques described above for the steganography application. In addition, since techniques of data hiding via texts sometimes produce illogical texts that are susceptible to human inspection, it is proposed to leverage the collaborative nature of office document editing to make covert communication more effective in face of steganalysis [28]-[30] and active warden attacks [31], which are discussed in more detail in Chapter 3.

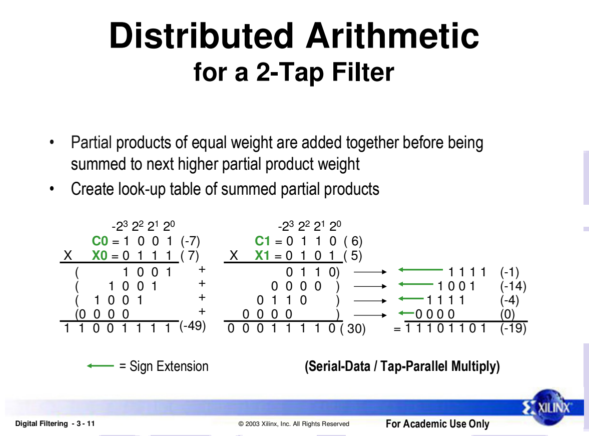
## 2.2 Data Hiding via Text Formatting and Layout

Office documents such as Microsoft Word documents allow very flexible formatting and layout of texts, including the precise controllability of text font sizes, colors, cases, styles, and effects; selection of various list and numbering options; flexible adjustability of inter-word, inter-line, and inter-paragraph spacings as well as line, tab, and paragraph indentations; setting of page and section margins; and so on.

It is possible to embed information into an office document by making small adjustments to the above-mentioned attributes in ways similar to those proposed for other media types. For example, Maxemchuk et al. [32]-[33] proposed to shift word and line spacings slightly (such as by 1/150 or 1/300 inch) in a document image to embed information; Zhong et al. [34] modified the spacing between characters within a line in a PDF to embed data; Villán et al [35] proposed to use color quantization to store data in electronic or printed documents; and Walton [36] described a technique of replacing the least-significant bits (LSBs) of the pixels of a cover image to embed information. Specifically, instead of shifting word or line spacings in a text image, we can modify the word or line spacing attributes in an office document slightly to embed information. And instead of changing the LSB values of pixel values, we could instead change the LSB values of the text color values in an office document.

Figure 2.2 shows an example of applying the technique of LSB replacement on text colors in an office document, where the word "Partial" in the first bullet-point in a slide is changed from completely black to a very dark gray. Such a modification is imperceptible, as seen in the left slide in the figure. The right slide in Figure 2.2 shows the result of applying *automatic style formatting* to the left slide, where the white background is changed into a dark blue one, and the black text color is changed into white. In this case, the data previously embedded using LSB replacement is still intact since the color remains unchanged as dark gray. However, the color modification is no longer imperceptible.

The challenge of using LSB replacement for data hiding via office documents in the presence of automatic style formatting as well as other attacks such as copying-and-pasting of contents are discussed in more detail in Chapter 6, and a novel technique is proposed for effective data hiding in slide presentations.



**Distributed Arithmetic**
**for a 2-Tap Filter**

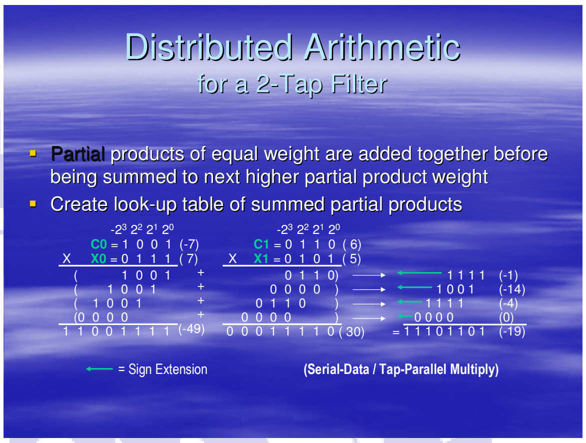
- Partial products of equal weight are added together before being summed to next higher partial product weight
- Create look-up table of summed partial products

```
        -2³ 2² 2¹ 2⁰              -2³ 2² 2¹ 2⁰
     C0 = 1  0  0  1 (-7)      C1 = 0  1  1  0 ( 6)
X    X0 = 0  1  1  1 ( 7)   X  X1 = 0  1  0  1 ( 5)
  (         1  0  0  1   +               0  1  1  0)  ──→  ←── 1 1 1 1    (-1)
  (      1  0  0  1      +            0  0  0  0   )  ──→  ←── 1 0 0 1     (-14)
  (   1  0  0  1         +         0  1  1  0      )  ──→  ←── 1 1 1 1      (-4)
  (0  0  0  0            +      0  0  0  0         )  ──→  ←── 0 0 0 0       (0)
1  1  0  0  1  1  1  1 (-49)  0  0  0  1  1  1  1  0 ( 30)  = 1 1 1 0 1 1 0 1 (-19)
```

← = Sign Extension **(Serial-Data / Tap-Parallel Multiply)**

Digital Filtering - 3 - 11 © 2003 Xilinx, Inc. All Rights Reserved **For Academic Use Only**

(a)



Distributed Arithmetic
for a 2-Tap Filter

- Partial products of equal weight are added together before being summed to next higher partial product weight
- Create look-up table of summed partial products

```
        -2³ 2² 2¹ 2⁰              -2³ 2² 2¹ 2⁰
     C0 = 1  0  0  1 (-7)      C1 = 0  1  1  0 ( 6)
X    X0 = 0  1  1  1 ( 7)   X  X1 = 0  1  0  1 ( 5)
  (         1  0  0  1   +               0  1  1  0)  ──→  ←── 1 1 1 1    (-1)
  (      1  0  0  1      +            0  0  0  0   )  ──→  ←── 1 0 0 1     (-14)
  (   1  0  0  1         +         0  1  1  0      )  ──→  ←── 1 1 1 1      (-4)
  (0  0  0  0            +      0  0  0  0         )  ──→  ←── 0 0 0 0       (0)
1  1  0  0  1  1  1  1 (-49)  0  0  0  1  1  1  1  0 ( 30)  = 1 1 1 0 1 1 0 1 (-19)
```

← = Sign Extension (Serial-Data / Tap-Parallel Multiply)

(b)

Figure 2.2. Illustration of slide designs. (a) A slide from a tutorial from Xilinx, Inc. with black texts on white background; (b) the slide in (a) with a slide design template of bluish background applied.

## 2.3 Data Hiding via Multimedia Contents

Office documents can contain an assortment of multimedia contents such as drawings, images, videos, and audios. Office software suites that are in common use today typically cannot manipulate audio or video contents, so these media are often stored as standalone files and an office document simply stores a *reference* to the external file. On the other hand, drawings and images are usually *embedded directly into* an office document for ease of manipulation, transmission, and storage. It is thus possible to achieve data hiding via office documents by applying existing data hiding techniques for drawings and images and then embed them into an office document.

For example, since the embedding of text images such as hand-written signatures or specially-styled headlines into an office document is a plausible scenario, we can apply techniques proposed for covert communication via text images [37]-[39] on these embedded images for the purpose of conveying a secret message via office documents. Such an approach is desirable as the sending of a hand-written signature by itself is relatively improbable compared to the case of embedding it in an office document. Also, steganalysis of a text image inside an office document is computationally more expensive than processing a stand-alone text image.

Compared to the cases of embedding text images, it is more common to embed various color images into an office document such as a slide presentation to illustrate or emphasize the key points mentioned in the document. One can thus use techniques proposed for embedding secret information into color images [40], [41] for the steganography application using a similar method as that mentioned previously. One may also use techniques proposed for embedding watermarks into images [42]-[48] for the copyright protection application by embedding watermarked images into an office document.

Digital watermarking methods for images are usually categorized into two types: *invisible* and *visible*[5]. The first type aims to embed copyright information imperceptibly into host media such that in cases of copyright infringements, the hidden information can be retrieved to identify the ownership of the protected host. It is important for the watermarked image to be resistant to common image operations to ensure that the hidden information is still retrievable after such alterations. Methods of the second type, on the other hand, yield visible watermarks which are generally clearly visible after common image operations are applied. In addition, visible watermarks convey ownership information directly on the media and can deter attempts of copyright violations.

Embedding of watermarks, either visible or invisible, degrades the quality of the host media in general. A group of techniques, named *reversible* watermarking [49]-[59], allow legitimate users to remove the embedded watermark and restore the original content as needed. However, not all reversible watermarking techniques guarantee *lossless image recovery*, which means that the recovered image is identical to the original, pixel by pixel. Lossless recovery is important in many applications where serious concerns about image quality arise. Some examples include forensics, medical image analysis, historical art imaging, or military applications.

Compared with their invisible counterparts, there are relatively few mentions of lossless visible watermarking in the literature. Several lossless invisible watermarking techniques have been proposed in the past. The most common approach is to compress a portion of the original host and then embed the compressed data together with the intended payload into the host [52]-[54]. Another approach is to superimpose the spread-spectrum signal of the payload on

[5] There is also the "cocktail" watermarking scheme [48] that embeds both types of watermarks simultaneously into an image, which makes it harder for an attacker to remove both types of watermarks.

the host so that the signal is detectable and removable [42]. A third approach is to manipulate a group of pixels as a unit to embed a bit of information [55]-[57]. Although one may use lossless invisible techniques to embed removable visible watermarks [51], [58], the low embedding capacities of these techniques hinder the possibility of implanting large-sized visible watermarks into host media.

As to lossless visible watermarking, the most common approach is to embed a monochrome watermark using deterministic and reversible mappings of pixel values or DCT coefficients in the watermark region [50], [59]. Another approach is to rotate consecutive watermark pixels to embed a visible watermark [59]. One advantage of these approaches is that watermarks of arbitrary sizes can be embedded into any host image. However, only *binary* visible watermarks can be embedded using these approaches, which is too restrictive since most company logos are colorful.

In Chapter 8, we describe a new method for lossless visible watermarking which allows the embedding of different types of visible watermarks into cover images, including the embedding of non-uniformly translucent full-color ones such as that illustrated in Figure 2.3 below. Such watermarks provide significantly better advertising effects than traditional monochrome ones when the images are embedded within office documents.

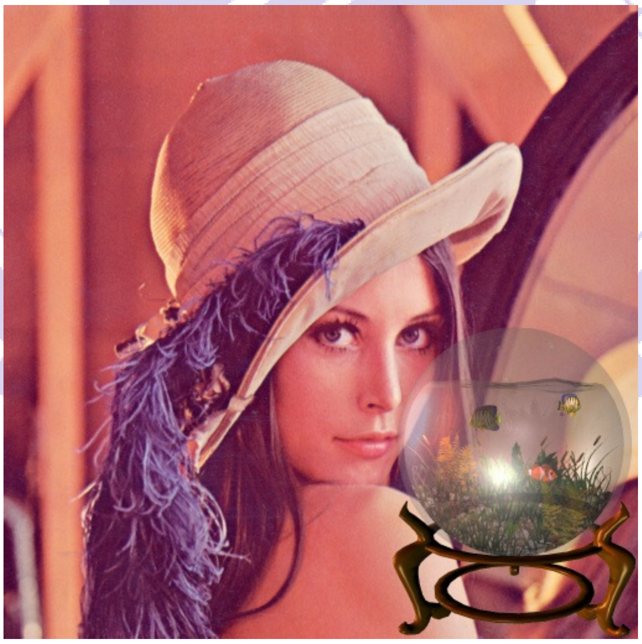


Figure 2.3. An image of Lena with a translucent full-color watermark “Globe” superimposed.

## 2.4 Data Hiding via Multimedia Formatting and Layout

In addition to hiding data inside the multimedia content themselves, it is also possible to leverage the formatting or the layout of the multimedia content embedded in an office document for data hiding applications. For example, images are often created in external programs and then embedded in office documents. For convenience, office application suites often allow these images to be adjusted, including their brightness and contrast values, size of appearances, amounts of cropping for the four edges, and positioning properties. Many of these formatting or layout properties may be used for various data hiding applications.

On the other hand, drawings are often created inside an office document using the office application software. Also, such drawings are usually *vector drawings* that contain objects of different shapes and sizes with uniform or gradient fills. Data hiding in vector drawings is comparatively less studied compared to data hiding in images, due to the relatively low information content in such a kind of media that can be manipulated.

Data hiding in a vector drawing is most commonly achieved by altering the geometry or positioning of the shapes in the drawing to embed data, the manipulation of which can be done in the spatial domain or in one of the transform domains such as DFT, DWT, and DCT [60]-[66]. Kwon et al. [61] embedded invisible watermark signals into lines, arcs, and circles in a CAD drawing by modifying their lengths, angles, and radii, respectively. Detection of the watermark, however, requires the use of the original drawing. Solachidis and Pitas [62] achieved blind watermark detection by modifying the coordinates of the vertices in a polygonal line using Fourier descriptors. The embedded watermark is resilient to scaling, rotation, and translation attacks, but vulnerable to distortion attacks. The method was later enhanced by Doncel et al. [63]. Im et al. [64] proposed the use of wavelet descriptors for embedding watermarks that are robust against global and local geometrical distortions.

It is noted that techniques that manipulate the internal coordinates of a shape itself cannot be applied to drawings such as flowcharts, network topologies, floor plans, and circuit diagrams, because objects in these diagrams come from *stencils*. Figure 2.4 shows an example of a floor plan drawing created in Microsoft Visio, where the shapes representing desks, chairs, servers, walls, doors, etc. all come from standard stencils, and cannot be individually altered. In Chapter 7, we describe how to manipulate the way that drawing objects are embedded in a Microsoft Visio drawing for data hiding applications.

Another technique for data hiding in multimedia formatting is that proposed by Yang and Chen [67], where the animation effects of objects in a Microsoft PowerPoint presentation are modified according to an animation codebook to embed a secret message for the steganography application. The work was later extended by Jing et al. [68] by further leveraging the animation timing effect variations for message embedding. One advantage of these techniques and that proposed in Chapter 7 is that the main content in the document is not distorted during message embedding. Another advantage is that these techniques can in general be used in conjunction with each other to extend the data hiding capacity as well as increase the complexity of steganalysis.

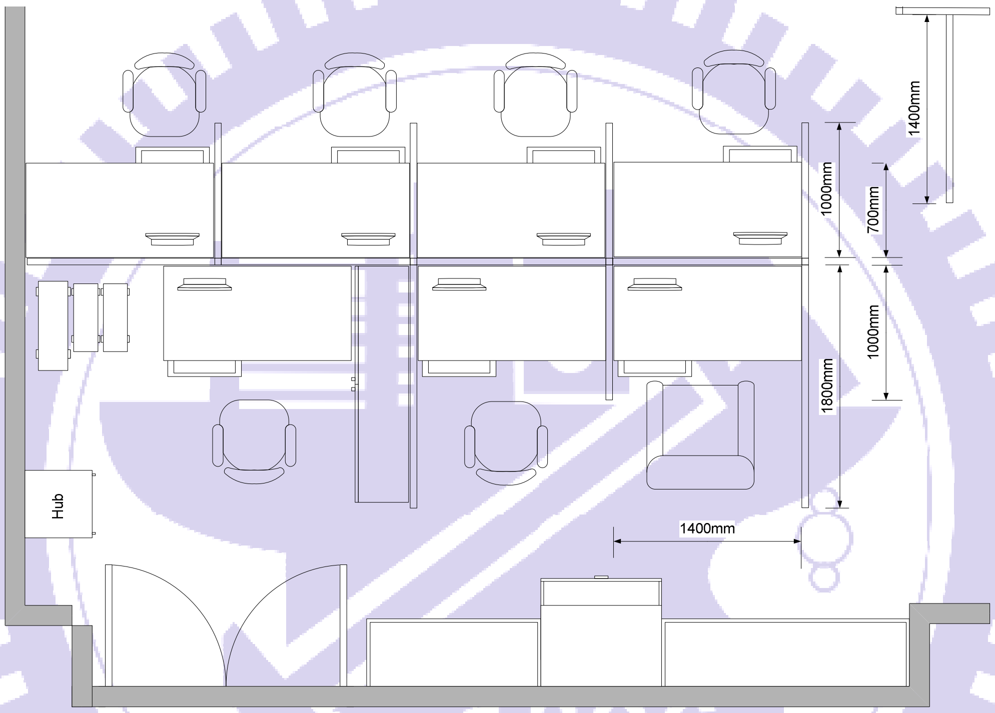


Figure 2.4. A floor plan diagram of an office composed of different objects from stencils.

## 2.5 Data Hiding via Auxiliary Data

Another approach to data hiding via an office document is simply to store information inside document metadata [69] such as the author, organization, description, and keyword fields that generally allow arbitrary information to be entered and stored. Liu et al. [70] proposed to store a secret message inside the *notes pages* of a Microsoft PowerPoint document. The embedding is made innocuous by generating the notes based on the sentences contained in the slides.

A type of interesting auxiliary data that can be embedded into an office document is program code, or *macro* [71]. Normal uses of macros can make document processing easier and more efficient [72], but it can also be used for new approaches to *active data hiding* [73].

However, since malicious codes such as viruses and worms can easily be embedded into macros, their uses are being limited by anti-virus software applications as well as the office applications themselves.

The technique of embedding information inside document auxiliary data is suitable for data hiding applications such as data association or media authentication (the technique is used in Chapter 4 and Chapter 5 for exactly these purposes), but is in general undesirable for applications such as copyright protection. This is because document metadata can usually be modified or removed easily without affecting the main content of the document, insofar as Microsoft has provided detailed how-to documents as well as tools [74], [75] for removing information embedded in the metadata of an office document.

## 2.6 Data Hiding via Physical File Formats

Data hiding via the physical file format of office documents has gained research traction recently, thanks to Microsoft's adoption of standardized file formats and opening-up of previous proprietary binary formats. One approach to data hiding via physical document files is to utilize unused spaces such as slack spaces at the end of *data streams* in a file [76] or redundant data that are created during consecutive file updates to a document [77], [78].

Another approach to data hiding via physical file formats is to exploit the *forward-compatible* nature of the document format, that is, application software will typically silently ignore unknown data blocks encountered while reading a file. Park et al. [79] described how unknown parts and unknown relationships in the Office Open XML documents (which is a zipped file containing XML documents and other supporting files[6]) can be used for steganography applications.

Finally, since the standard-based office document formats such as Office Open XML and OpenDocument are (compressed) XML files, one may use data hiding techniques proposed for XML files on such documents. For example, the five techniques proposed by Inoue et al. [80] for embedding data into XML documents may be applied to office documents for data hiding applications: 1) alternate representation of empty elements; 2) use of white spaces in tags; 3) utilizing the order of appearance of elements; 4) utilizing the order of appearance of attributes; and 5) alternate representation of elements that can contain other elements.

[6] This is also true for the OpenDocument format.

## 2.7 Summary

In this chapter we presented six areas for data hiding via office documents and point out related works that can be used for data hiding in each area. The first five areas (i.e., data hiding in text, data hiding in text formatting and layout, data hiding in multimedia content, data hiding in multimedia formatting and layout, and data hiding in auxiliary data) can be regarded as data hiding in the logical regions of the office document. These techniques are in general more resilient to common operations performed on office documents compared to techniques that exploit the physical file format directly. One reason is that an office software application has no obligation to preserve the content and structure of non-user-visible data. This is especially true in file format conversions, such as converting a file between the Office Open XML and the OpenDocument formats, where unknown and hence unconvertible contents are simply discarded.

In the subsequent chapters, we focus on data hiding in the logical areas of the office document (i.e., data hiding via texts, data hiding via text formatting and layout, data hiding via multimedia contents, data hiding via multimedia formatting and layout, and data hiding via auxiliary data) as opposed to hiding in the physical file formats. Experimental results are included to demonstrate that the proposed data hiding techniques can indeed embed data that survive attacks such as file format conversions and common editing operations. A summary of the areas utilized for data hiding via office documents for each of the proposed method can be found in Table 9.1 in the last chapter.

# Chapter 3
# A New Steganographic Method for Data Hiding in Microsoft Word Documents by a Change-Tracking Technique

## 3.1 Introduction

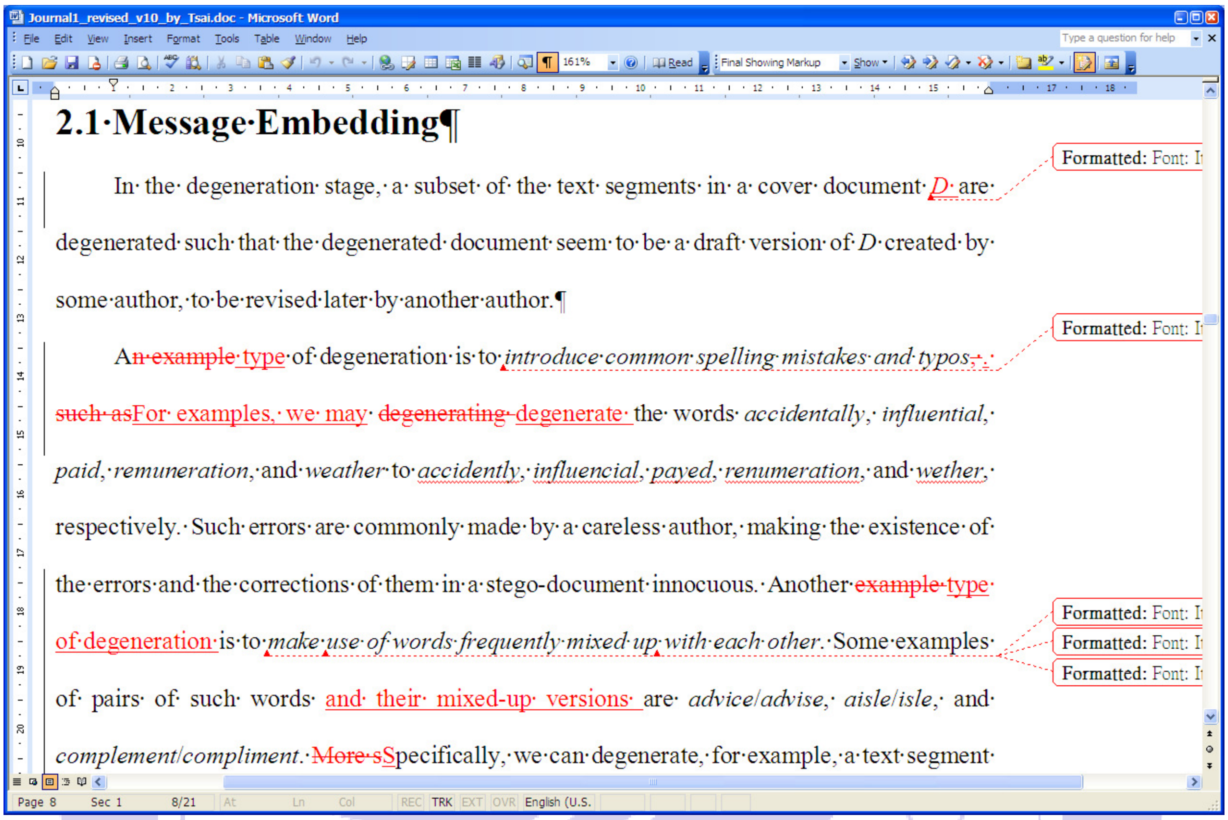
Office documents are sometimes written by multiple authors who may be physically distant from each other. To facilitate communications between the authors during the collaborative document authoring process, a word processor such as Microsoft Word can be used to record the exact modifications performed by an author and embeds the ways of revisions as *change-tracking information* into the document. From such change-tracking information, one can discern the exact changes made by a prior author, and can recover a prior version of the document if necessary.

Figure 3.1 shows an example of the collaborative document authoring process in Microsoft Word, where an author is modifying a document and the word processor is tracking the author's modifications. The modifications by the author are clearly marked, with the deleted words stroked-through and newly inserted text underlined[7]. Formatting changes are displayed as comment bubbles at the right side margin of the page. Each collaborating author can accept or reject each of the modifications made by another author. It is a common practice for a collaborating author to review and then accept or reject each of the modifications in a document first before performing his/her own corrections.

We have chosen Microsoft Word documents as cover media, which provide change-tracking facilities to materialize the proposed method. Communications via Word documents are commonplace for personal, business, or academic purposes nowadays, so transmissions of such documents will not be under close scrutiny. We note that any other document format that offer change-tracking facilities, such as OpenDocument, can also be

[7] There are commands and options to change the ways the modifications are displayed, for example showing deletions and insertions as comment bubbles at the side or listing the modifications line-by-line in a separate panel.

used for data hiding using the proposed method, though for simplicity we shall limit subsequent discussions to Microsoft Word documents only.


Journal1_revised_v10_by_Tsai.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Type a question for help
161%
Read
Final Showing Markup
Show
Formatted: Font: I
Formatted: Font: I
Formatted: Font: I
Formatted: Font: I
Formatted: Font: I
Page 8 Sec 1 8/21 At Ln Col REC TRK EXT OVR English (U.S.

2.1 Message Embedding¶

In the degeneration stage, a subset of the text segments in a cover document *D* are degenerated such that the degenerated document seem to be a draft version of *D* created by some author, to be revised later by another author.¶

A~~n example~~ type of degeneration is to *introduce common spelling mistakes and typos*~~,~~. ~~such as~~For examples, we may ~~degenerating~~ degenerate the words *accidentally*, *influential*, *paid*, *remuneration*, and *weather* to *accidently*, *influencial*, *payed*, *renumeration*, and *wether*, respectively. Such errors are commonly made by a careless author, making the existence of the errors and the corrections of them in a stego-document innocuous. Another ~~example~~ type of degeneration is to *make use of words frequently mixed up with each other*. Some examples of pairs of such words and their mixed-up versions are *advice*/*advise*, *aisle*/*isle*, and *complement*/*compliment*. ~~More s~~Specifically, we can degenerate, for example, a text segment

Figure 3.1. Screenshot of Microsoft Word in a case of collaborative document authoring.

## 3.2 Overview and Merits of Proposed Method

In this chapter we describe the proposed technique to embed a secret message into a Microsoft Word document by using change-tracking information. The basic idea of the proposed method is to *degenerate* the contents of a cover document $D$ to arrive at another document $D'$ by embedding a secret message $M$ in $D$ during the transformation process, as shown in Figure 3.2. The degeneration process introduces errors into the *degenerated document* $D'$ (or performs semantically-equivalent transformations), such that the degenerated document appears to be a preliminary work by a virtual author $A'$, which is to be revised later by another author $A$. A *stego-document* $S$ is then produced from $D'$ by revising $D'$ back to $D$ with the changes being tracked, making it appear as if author $A$ is correcting the errors in $D'$. On the other hand, by making use of the change-tracking information in the stego-document $S$, a recipient $B$ of $S$ can easily recover the original document $D$ as well as the degenerated document $D'$, from both of which the embedded message can be extracted.

In more detail, the cover document $D$ is partitioned into *text segments* $d_1$, $d_2$, …, and $d_n$, and each segment $d_i$ is either kept unchanged or degenerated into a new version during the degeneration stage to arrive at a degenerated document $D'$ containing *degenerated text segments*, $d'_1$, $d'_2$, …, $d'_n$. The secret message is embedded during the degeneration process. In the revision stage, each previously degenerated text segment $d'_i$ is revised back to $d_i$ with the revisions made being tracked using the "Track Changes" feature of Microsoft Word, resulting in a final stego-document $S$ consisting of *revised text segments* $s_1$, $s_2$, …, and $s_n$. Here each $s_i$ includes its original text segment $d_i$ and the associated change-tracking information.

As an example, suppose the text segment $d_5$ = "travel" was degenerated to be $d'_5$ = "move," the revised text segment $s_5$ might be seen as "~~move~~travel" in Microsoft Word. To an outside observer, it appears as if $A'$ has originally written "move," but it was corrected to be "travel" by $A$.

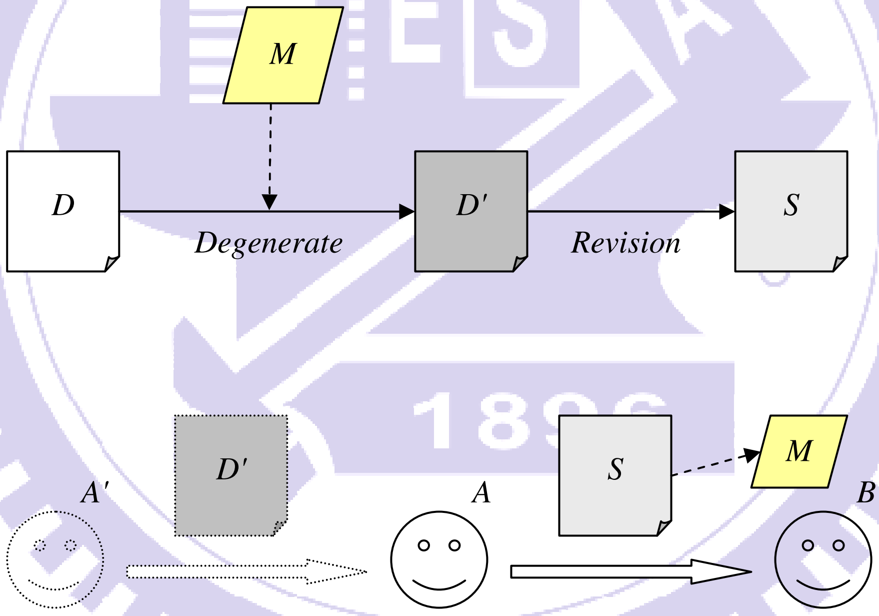


Figure 3.2. Author $A$ sends a stego-document $S$ with embedded message $M$ to a recipient $B$ after embedding $M$ into a cover document $D$ to form $S$ that appears to be the collaborative product of multiple authors $A$ and $A'$.

We describe below some advantages of the proposed approach over previous methods such as those described in Section 2.1. First, communication via a Word document, especially for collaborative authoring, is a common and natural scenario and hence will not be under close scrutiny.

Next, the use of known cover media is in general impossible with most prior steganography techniques because discrepancies between the cover medium and the corresponding stego-medium are generally illogical. For example, if the synonym replacement technique [15] was used on known cover text, an adversary can simply compare the original and the stego-text and easily conclude that the synonym replacement technique was used. On the other hand, the proposed method provides legitimate cases in using a known cover document. For example, an already published document that is collaboratively authored can be used as a cover document. The stego-document *S* appears to be the version of the paper before change-tracking information removal and submission for publication. The transmission of *S* by one of the collaborating authors to another author, a colleague, or a supervised student of the author is reasonable – a colleague or a student receiving the document containing the change-tracking information can learn of the mistakes made by a colleague and the appropriate corrections thereof.

Linguistic steganography methods that generate the cover text directly, such as that described by Wayner [12], do not have the known cover problem. However, the text produced using these methods are often implausible. Similarly, linguistic steganography methods [11] that manipulate cover text by performing semantically-equivalent transformations also often produce contents that are easily detectable by a human reader. This is especially true if the characteristics of the sender is known (e.g., if *A* is a well-known Professor), meaning the content produced via text manipulations will easily cause suspicion to readers familiar with the sender's work. Many of the prior techniques, however, can be used to a greater effect in the proposed method in degenerating a cover document *D* to another *D'*. The inconsistencies and errors in *D'* are more tolerable in this setting because *D'* is disguised to be the draft work of an inferior author.

Most of the proposed steganography techniques embed secret information in the *subliminal channel* of a cover medium, that is, the unnoticeable, redundant, or useless parts of a cover medium are manipulated to embed secret data. These techniques are vulnerable, however, in the presence of active wardens, who are allowed to introduce subtle modifications to passing stego-objects between two parties, as long as the modifications do not interfere with normal communications of the parties. For example, if information is hidden in HTML files by adding useless spaces and line breaks or special codes [81], [82], an active warden can simply remove all redundant spaces and special codes in a passing HTML file because he is certain that such actions will not affect normal communications, while any information hidden that could have been in the file are removed. Similarly, if information is hidden in

HTML files by changing the case of letters in the tags [83], an active warden can simply alter all letters in the tags to be lower case to remove all possible hidden communications. In contrast, the proposed method embeds secret data in the change-tracking information, which are vital in collaborative document authoring and cannot be tampered with ignorantly.

## 3.3 Proposed Message Embedding Process

During the proposed message embedding process, a subset of the text segments in the cover document is degenerated. The indices of the chosen text segments are called the *embedding places*, denoted as a set by $P = \{i_1, i_2, \ldots, i_G\}$, where $G$ is the number of text segments degenerated and $1 \leq i_k \leq n$ for $1 \leq k \leq G$.

One type of degeneration is to introduce common spelling mistakes and typos. For example, we may degenerate the words accidentally, influential, paid, remuneration, and weather to accidently, influencial, payed, renumeration, and wether, respectively. Such errors are commonly made by a careless author, making the existence of the errors and the corrections of them in a stego-document innocuous.

Another type of degeneration is to make use of words frequently confused with each other. Some examples of pairs of such words and their mixed-up versions are advice/advise, aisle/isle, and complement/compliment. Specifically, we can degenerate, for example, a text segment $d$ = "advice" in the cover document into $d'$ = "advise" by using the pair of words advice/advise.

There are also many other types of degenerations, such as synonym replacements, syntactic transformations, and other techniques from linguistic steganography studies reviewed in Section 2.1. As mentioned previously, the use of existing degeneration techniques in the proposed framework makes the resulting imperfect text less conspicuous.

In general, we define a *degeneration set* $R_d$ to be the ordered set of possible degenerated text segments for a text segment $d$. We use the notation $R_d(j)$ to denote the $j^{\text{th}}$ element in $R_d$, and the notation $\Pr\{R_d(j)\}$ to denote the probability of occurrence for $R_d(j)$. The probabilities of occurrences are used during message embedding so that the system prefers common substitutions and thus produces a more natural stego-document. It is noted that $\Pr\{R_d(1)\} + \Pr\{R_d(2)\} + \ldots + \Pr\{R_d(c)\} = 1$, where $c = |R_d|$, the size of $R_d$.

With the previously-defined notations, the text segments $d_i$, $i \in P$, are individually degenerated to be $R_{d_i}(j_i)$, $1 \leq j_i \leq |R_{d_i}|$, with the degeneration indices $j_i$ dependent on the message $M$ being embedded.

The message is treated as an $m$-bit bit string $M = b_1b_2...b_m$, where each $b_i$ is a bit. A message length header $H$ is added in front of $M$ and strings of random 0's and 1's are padded to the end of $M$ as necessary during the message embedding. That is, we embed into the cover document the bit string $M' = l_1l_2...l_Lb_1b_2...b_mx_1x_2...$, with $H = [l_1\ l_2\ ...\ l_L]$ being the message length header with the value $m$, and $x_i$ being the padding bits that are selected randomly. The communicating parties should agree on the magnitude of $L$ beforehand, such that it can accommodate the longest message that is to be communicated between the parties.

To face the common assumption made in covert communication that the data hiding algorithms used may be known to the public, it is suggested to randomize the secret message in advance by some symmetric encryption algorithm such as AES [84] before it is taken as input to the proposed data embedding process.

The message bits are embedded using Huffman coding at each embedding place in a way similar to that proposed by Wayner [13]. The previously-mentioned probabilities of occurrences of $R_d(j)$ are used to assign variable-length Huffman codes to different degenerations. Shorter Huffman codes are assigned to degenerations with higher probabilities of occurrences and longer ones to those with lower probabilities of occurrences. A degeneration with a shorter Huffman code is more likely to match the message bits being embedded and hence more possible to be selected as the choice of degeneration. The details of the message embedding process are presented in the algorithm below.

**Algorithm 3.1: embedding a message by text degeneration and revision.**

***Input*:** a cover document $D$ partitioned into text segments $d_1, d_2, ..., d_n$; a message to be embedded $M' = l_1l_2...l_Lb_1b_2...b_mx_1x_2... = b'_1b'_2b'_3...$; and a secret key $K$.

***Output*:** a stego-document $S = \{s_1, s_2, ..., s_n\}$.

***Steps*:**

1. Initialize the set $P$ of embedding places to be empty.
2. Define an index $p$ pointing to the position of the message bit $b'_p$ which we are currently encoding, with initial $p = 1$.
3. Select an embedding place $i$ randomly using $K$ such that $i$ is in the range of $1 \leq i \leq n$ and not in the set $P$; and then add $i$ to $P$.
4. Construct as follows a Huffman tree $T$ for the text segment $d_i$ with degeneration set $R_{d_i}$ of size $c$.

   a. Create leaf nodes $n_1, n_2, \ldots, n_c$, and assign a weight of $w_k = \Pr\{R_{d_i}(k)\}$ to node $n_k$ for all $1 \le k \le c$.
   b. Initialize a set $Q$ to contain all the leaf nodes $n_1, n_2, \ldots, n_c$.
   c. Find in $Q$ the node $n'$ with the minimum weight $w'$ and the node $n''$ with the second smallest weight $w''$; and then remove $n'$ and $n''$ from $Q$.
   d. Create a new node $\eta$ with weight $w' + w''$, and assign $n'$ as its left child and $n''$ as its right child.
   e. If $Q$ is empty, then tree $T$ has been constructed and take $\eta$ as its root; else, add node $\eta$ to $Q$ and go to Step 4c.
5. Degenerate text segment $d_i$ to be $d_i' = R_{d_i}(j)$, where the degeneration choice $j$ is determined as follows.
   a. Starting from the root of tree $T$, traverse $T$ to the left child if $b'_p$ is 1 or to the right child if $b'_p$ is 0.
   b. Increment $p$ and continue node traversal in a similar way until a leaf node $n_j$ is reached.
   c. Take the index $j$ of $n_j$ as the desired degeneration choice.
6. Repeat Steps 3 through 5 until the entire message has been embedded, that is, until $p > L + m$.
7. Revise each previously degenerated text segment $d'_i$ back to $d_i$ with the revisions made being tracked to yield stego-text $s_i$ for all $i$ in $P$.

We note that the proposed embedding procedure is generic with no restriction imposed on the elements in a degeneration set $R_d$. In our experiments, we used a common English errors database, a synonym database, and a collection of real-world collaborative editing entries to construct the degeneration database.

As an example, suppose that the degeneration set $R_d$ for $d$ = "travel" contains the seven entries *go*, *travel*, *trip*, *journey*, *jaunt*, *locomote*, and *move*, with the respective probabilities of occurrences as shown in Table 3.1. Then steps 4 and 5 of Algorithm 3.1 will essentially generate the Huffman tree shown in Figure 3.3. The text $d$ in a document $D$ is degenerated to one of the seven choices depending on the message being embedded. In the case that the message to be embedded is 1100, $d$ will be degenerated as $d'$ = "trip." Finally, $d'$ is revised back to $d$ with the changes being tracked to yield the stego-text $s$ = "~~trip~~travel."

Table 3.1. Occurrence probabilities and Huffman codes for the entries in an example degeneration set of "travel."

| $j$ | $R_d(j)$ | $\Pr\{R_d(j)\}$ | Huffman code |
|---|---|---|---|
| 1 | go | 0.6306889 | 0 |
| 2 | travel | 0.2118223 | 10 |
| 3 | trip | 0.0536723 | 1100 |
| 4 | journey | 0.0273936 | 11010 |
| 5 | jaunt | 0.0004189 | 110110 |
| 6 | locomote | 0.0000347 | 110111 |
| 7 | move | 0.0759693 | 111 |

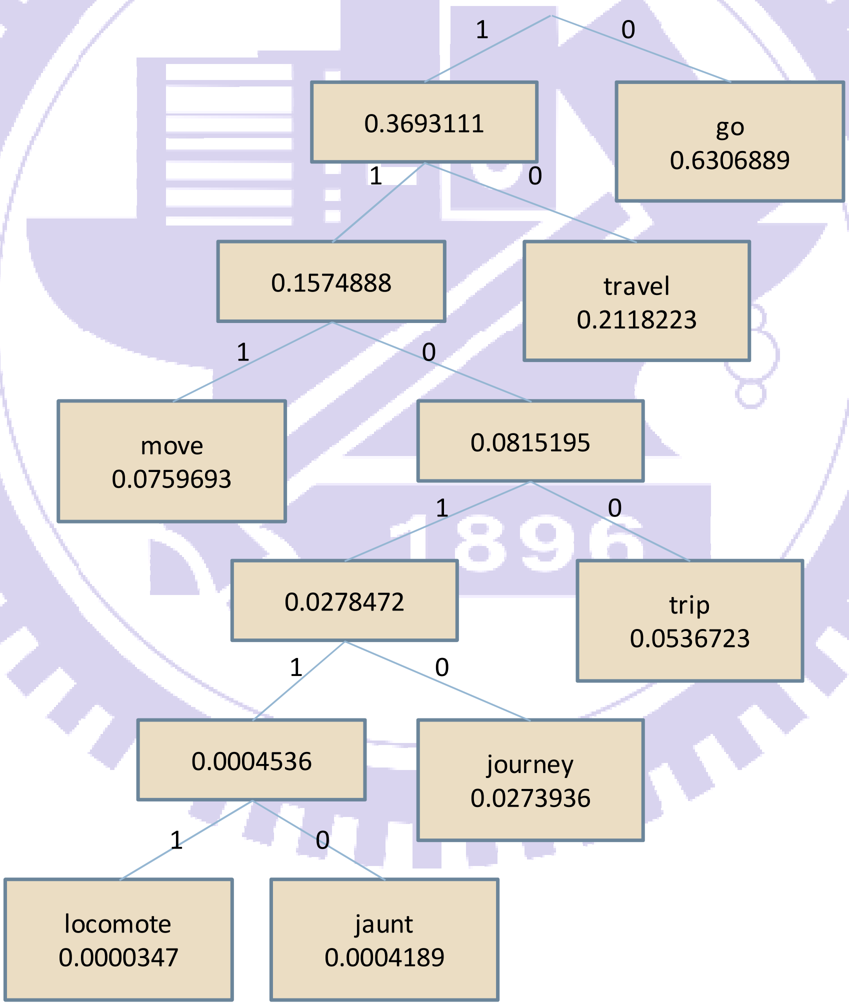


Figure 3.3. Huffman tree constructed by Algorithm 3.2 for the entries listed in Table 3.1.

## 3.4 Proposed Message Extraction Process

The change-tracking information included in the stego-document $S$ allows simple recovery of the original document $D$ and the degenerated document $D'$, from both of which the embedded message can be extracted. The proposed message extraction method is described in the following algorithm.

**Algorithm 3.2: extracting a message by text degeneration and revision.**

***Input*:** a stego-document $S = \{s_1, s_2, ..., s_n\}$ and a secret key $K$.

***Output*:** the extracted message $M' = b'_1b'_2b'_3 = l_1l_2...l_Lb_1b_2...b_mx_1x_2....$

**Steps:**

1. Recover the original documents $D = \{d_1, d_2, ..., d_n\}$ and the degenerated document $D' = \{d'_1, d'_2, ..., d'_n\}$ from $S$ by using the change-tracking information and the related operations provided by Microsoft Word.
2. Initialize the set of embedding places $P$ to be empty.
3. Define an index $p$ pointing to the position of the message bit $b'_p$ which we are currently decoding, with initial $p = 1$.
4. Select the same embedding place $i$ as that in message embedding using $K$ and $P$; and then add $i$ to $P$.
5. Construct a Huffman tree $T$ for the text segment $d_i$ with degeneration set $R_{d_i}$ in the same way as that in message embedding.
6. Determine the choice of degeneration $j$ such that $R_d(j) = d'_i$.
7. Decode the message bits encoded in $j$ in the following way:
   a. Starting from the root of the tree $T$, traverse it to the leaf node $n_j$ and note the path traversed.
   b. Analyze the path traversed, and set $b'_p$ to be 1 if the path goes down a left child; or to be 0 if the path goes down the right. Increment the value of $p$ for each child traversed.
8. Repeat Steps 4 through 7 until the entire message has been extracted, that is, until $p > L + m$.

As an example, given a revised text segment $s$ = "~~move~~travel," we can recover the original and the degenerated text segments to be $d$ = "travel" and $d'$ = "move," respectively. Suppose that the degeneration set $R_d$ is the same as that in the example of Algorithm 3.1 and hence Step 5 of Algorithm 3.2 will construct the same Huffman tree shown in Figure 3.3.

Since the degenerated text segment is “move,” we can conclude that the bits “111” were previously embedded.

## 3.5 Experimental Results

The proposed method was implemented[8] using Microsoft C$^{\#}$.NET and Microsoft Office 2003, and the Automation technique provided by Microsoft was used to manipulate Word documents. The degeneration sets were constructed by using public linguistic databases as well as real past collaborative editing files kept by Dr. Tsai.

The first degeneration database was constructed by using entries from the list of common errors in English compiled by Brians [85]. We filtered out entries that are automatically corrected by Microsoft Word by its *AutoCorrect* feature (such as parallelled, therefor, etc.), as these are unsuitable candidates for degeneration. Table 3.2 summarizes the resultant database used, which contains 760 degeneration sets in total. We have included the text $d$ itself in the set $R_d$, such that if it is chosen as the degeneration choice, the text is effectively *not* degenerated.

Table 3.2. Summary of common English errors database used as $R_d$.

| $\lvert R_d \rvert$ | Examples of $d$ and $d'$ | Entries |
|---|---|---|
| 2 | breach (breech), criteria (criterion), loose (lose), ordnance (ordinance) … | 675 |
| 3 | palette (palate, pallet), imminent (eminent, immanent), … | 71 |
| 4 | morale (ethics, morals, moral), carat (caret, carrot, karat), … | 14 |

We have also utilized the WordNet 2.1 lexical database [86] to obtain sets of synonyms for the degeneration purpose. The occurrence probabilities of the entries in $R_d$ for both the common English errors and the synonyms are estimated by making use of the Google SOAP Search API [87][9]. Specifically, we query the Google web index to get the approximate number

[8] The implementation can be downloaded at http://sites.google.com/site/ktyliu/data-hiding-in-microsoft-word-by-change-tracking.

[9] The Google SOAP Search API is unfortunately no longer available for public use at the time of this writing. One simple alternative is to analyze a large public text copora and count the occurrences instead of querying the web.

of web pages that contain an entry in $R_d$. After the occurrences of each entry in $R_d$ are determined, the occurrence probabilities for the entries can be calculated. As an example, for $d$ = “study,” we use WordNet to find its synonyms “report,” “subject,” “work,” “field,” “survey,” “discipline,” “sketch,” “bailiwick,” and “cogitation.” Table 3.3 shows the approximate number of pages in Google web search, the occurrence probabilities, and the resulting Huffman codes for these words.

Table 3.3. Occurrence probabilities and Huffman codes of “study” and its synonyms.

| **Word** | **Pages on Google** | **Occurrence probability** | **Huffman code** |
|---|---|---|---|
| report | 1560000000 | 0.1733984 | 000 |
| subject | 1180000000 | 0.1311603 | 001 |
| work | 2630000000 | 0.2923319 | 01 |
| field | 1850000000 | 0.2056327 | 10 |
| survey | 838000000 | 0.0931461 | 111 |
| study | 758000000 | 0.0842538 | 1100 |
| discipline | 108000000 | 0.0120045 | 11010 |
| sketch | 71500000 | 0.0079474 | 110110 |
| bailiwick | 775000 | 0.0000861 | 1101110 |
| cogitation | 348000 | 0.0000387 | 1101111 |

We have also collected files of Dr. Tsai’s editing of his students’ works, including 157 thesis chapters and 20 journal and conference papers, in recent years. The collaborative editing entries were collected by first identifying sentences in the documents that contain change-tracking information. The text before and that after editing were then extracted, and identical starting and ending phrases in the two text segments are trimmed away so that the entry can be more widely applicable. Finally, we group the entries according to the text after editing. The database constructed using these real-life collaboration entries are then used during the degeneration stage. The degenerations made by using this database are realistic and the resulting stego-documents are indistinguishable from a document in real collaborative editing scenarios. For simplicity, we name the common errors in English database, the synonym database, and the real collaboration entries database as databases 1, 2, and 3, respectively in the subsequent discussions.

Figure 3.4 shows some extracts of the stego-documents produced using the proposed method by using a combination of databases 1 and 3, and Figure 3.5 shows some extracts of the stego-documents produced by using a combination of databases 1 and 2. We observe that the ways of degenerations are distinctively different when using different databases.

A set of experiments are then conducted to quantitatively measure the message embedding capacity of the proposed method when using the above databases. In more details, five theses and eight journal and conference papers are selected as test documents, and the numbers of words and sentences of the documents are listed in Table 3.4. The maximum message length that is embeddable for a document is message-dependent due to the Huffman coding used in the message embedding. In this set of experiments, we embedded random 0's and 1's into all embeddable places in the test document and repeated the embedding ten times for each test document. The average bits embeddable for each document are shown in Table 3.4. The increases in file size after the embedding was also measured and recorded in Table 3.4.

Table 3.4. Experimental results of message embedding capacity and increase in file size.

| Document | Sentences / Words | Databases 1 & 2 | | Databases 1 & 3 | |
|---|---|---|---|---|---|
| | | Capacity (bits) | File size increase (bytes / percentage) | Capacity (bits) | File size increase (bytes / percentage) |
| Paper 1 | 240<br>4075 | 1,558 | +415,744<br>+308% | 261 | +198,144<br>+147% |
| Paper 2 | 351<br>4903 | 1,768 | +419,328<br>+244% | 374 | +245,760<br>+143% |
| Paper 3 | 143<br>2596 | 782 | +158,208<br>+15% | 181 | +85,504<br>+8% |
| Paper 4 | 287<br>3518 | 1,163 | +265,216<br>+253% | 207 | +119,808<br>+114% |

Table 3.4 (cont.). Experimental results of message embedding capacity and increase in file size.

| | | | | | |
|---|---|---|---|---|---|
| Paper 5 | 223<br>4341 | 1,421 | +262,144<br>+261% | 321 | +128,512<br>+128% |
| Paper 6 | 222<br>3857 | 1,241 | +306,688<br>+216% | 301 | +159,744<br>+113% |
| Paper 7 | 350<br>5545 | 2,141 | +457,216<br>+367% | 470 | +269,312<br>+216% |
| Paper 8 | 338<br>8978 | 2,178 | +489,472<br>+296% | 451 | +176,128<br>+107% |
| Thesis 1 | 1204<br>15414 | 4,121 | +924,672<br>+237% | 1,605 | +869,888<br>+188% |
| Thesis 2 | 1415<br>20547 | 7,282 | +1,443,840<br>+312% | 1,620 | +954,880<br>+224% |
| Thesis 3 | 1213<br>18918 | 6,585 | +1,478,656<br>+347% | 1,750 | +1,061,376<br>+183% |
| Thesis 4 | 1456<br>21903 | 7,276 | +1,614,336<br>+279% | 1,477 | +439,808<br>+112% |
| Thesis 5 | 1150<br>20254 | 6,773 | +1,009,664<br>+257% | 1,145 | +691,200<br>+177% |

On average, 0.33 bits can be embedded into each word and 5.42 bits can be embedded into each sentence when using a combination of databases 1 and 2, and 0.07 bits per word and 1.19 bits per sentence can be embedded when using a combination of databases 1 and 3. On average, the stego-document increases by 21.7 bytes for each bit embedded when using databases 1 and 2, and by 53.8 bytes for each bit embedded when using databases 1 and 3. The embedding capacity is higher when using databases 1 and 2 because the WordNet database used produced many synonyms for a text segment. This in turn also lowers the increase in file size for each bit embedded.

## 3.6 Security Considerations

In the proposed method, it is assumed that the degeneration sets and the key used are agreed upon by the sending and receiving parties beforehand. The degenerations in the degeneration database should model realistic errors to counter visual steganalysis. Our way of using existing collaborative data as done in the experiments described previously can achieve this purpose. Specifically, an adversary inspecting a stego-document yielded by our experiments could not tell whether it is really an actual author making the mistakes, or whether the mistakes are introduced for the steganographic purpose by using the proposed method.

Next, it is noted that the proposed approach is robust against statistical steganalysis because the degenerations are chosen according to their occurrence probabilities, resulting in occurrence frequencies of degenerations in a stego-document being in line with those of normal documents. To ensure the statistical properties of the degenerations of a stego-document is close to that of a normal document, we can encrypt the message first before embedding so that the bits of the encrypted message are randomly distributed. Nevertheless, there is still a chance for the statistics of the degenerations to stray away from that of a normal document. To ensure maximal statistical coherence, we can alter the occurrence probabilities of the degenerations appropriately during message embedding. For example, we can halve the occurrence probability of a degeneration option after it has been chosen so that it is less likely to be chosen later again, thus achieving the desired statistical coherence with a normal document.

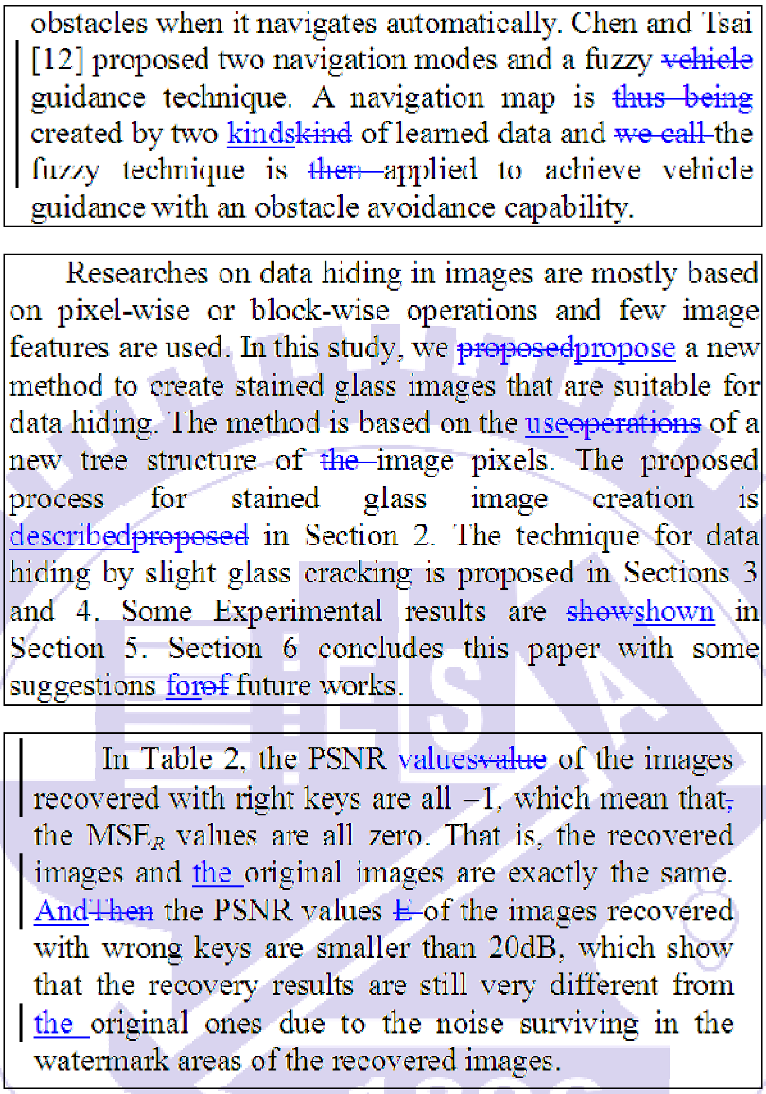
In addition, to ensure that the scheme is as inconspicuous as possible to adversaries, the degeneration database used should only be known by the communicating parties. This can be achieved for example by using an *evolving* degeneration database by modifying the degeneration database dynamically when normal collaborative documents are transmitted between the parties. One way to implement this idea is for the communicating parties to add into the degeneration database a key-dependent subset of the degenerations derived from collaboratively working on a normal document. In this way, an adversary cannot determine the exact contents of the degeneration database being used.

Another technique that can be employed is for a sender to use the proposed method to embed the intended payload into a stego-document and then manipulate the unused portions (essentially the padding parts of $M'$) of the stego-document to include degenerations outside their agreed degeneration database to mislead adversaries. The extra degenerations so introduced are assumed to be ignored by the receiver.

## 3.7 Summary

A new steganographic method for data hiding in Microsoft Word documents have been presented in this chapter. The data embedding is disguised such that the stego-document appears to be the product of a collaborative writing effort. Information is embedded in the degeneration stage of document transformation with steganographic effects. The degeneration stage introduces different degenerations mimicking an author with inferior writing skill, with the secret message embedded in the choices of degenerations using Huffman coding. The proposed message embedding and extraction methods have been implemented, proving the feasibility of the proposed method.

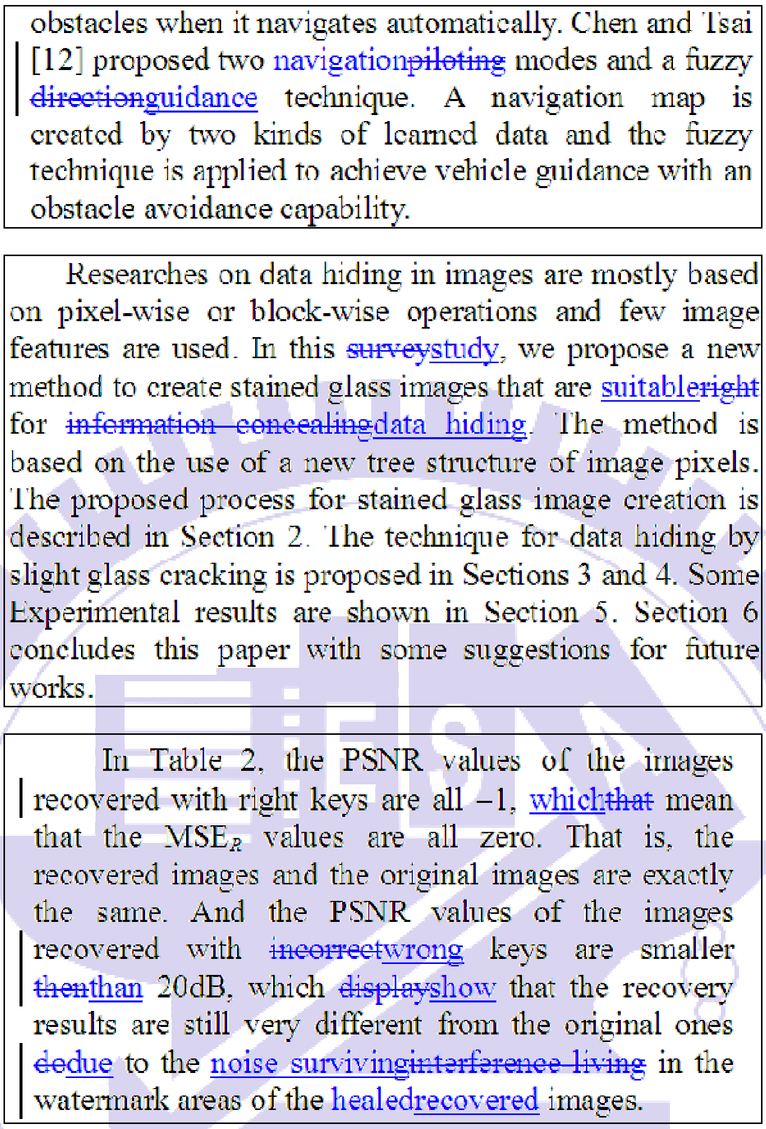
The proposed change-tracking technique for the steganography purpose is special in that the modifications made during embedding are essential information not to be tampered with ignorantly. On the contrary, methods that are based on redundant or unused information, imperceptibility, or alternative representations, are more vulnerable against attacks by an active warden, who can process all suspects without affecting normal cases of usage. The degeneration database used during degeneration does not need to be private to the sender and the receiver if the degenerations are realistic, as the warden cannot distinguish between legitimate collaboration cases and covert communication cases.


obstacles when it navigates automatically. Chen and Tsai [12] proposed two navigation modes and a fuzzy ~~vehicle~~ guidance technique. A navigation map is ~~thus being~~ created by two kinds~~kind~~ of learned data and ~~we call~~ the fuzzy technique is ~~then~~ applied to achieve vehicle guidance with an obstacle avoidance capability.

Researches on data hiding in images are mostly based on pixel-wise or block-wise operations and few image features are used. In this study, we ~~proposed~~propose a new method to create stained glass images that are suitable for data hiding. The method is based on the use~~operations~~ of a new tree structure of ~~the~~ image pixels. The proposed process for stained glass image creation is described~~proposed~~ in Section 2. The technique for data hiding by slight glass cracking is proposed in Sections 3 and 4. Some Experimental results are ~~show~~shown in Section 5. Section 6 concludes this paper with some suggestions for~~of~~ future works.

In Table 2, the PSNR values~~value~~ of the images recovered with right keys are all −1, which mean that~~,~~ the MSE$_R$ values are all zero. That is, the recovered images and the original images are exactly the same. And ~~Then~~ the PSNR values ~~E~~ of the images recovered with wrong keys are smaller than 20dB, which show that the recovery results are still very different from the original ones due to the noise surviving in the watermark areas of the recovered images.

Figure 3.4. Extracts of stego-documents produced using the proposed method with databases 1 and 3.


obstacles when it navigates automatically. Chen and Tsai [12] proposed two navigation~~piloting~~ modes and a fuzzy ~~direction~~guidance technique. A navigation map is created by two kinds of learned data and the fuzzy technique is applied to achieve vehicle guidance with an obstacle avoidance capability.

Researches on data hiding in images are mostly based on pixel-wise or block-wise operations and few image features are used. In this ~~survey~~study, we propose a new method to create stained glass images that are suitable~~right~~ for ~~information concealing~~data hiding. The method is based on the use of a new tree structure of image pixels. The proposed process for stained glass image creation is described in Section 2. The technique for data hiding by slight glass cracking is proposed in Sections 3 and 4. Some Experimental results are shown in Section 5. Section 6 concludes this paper with some suggestions for future works.

In Table 2, the PSNR values of the images recovered with right keys are all −1, which~~that~~ mean that the MSE$_R$ values are all zero. That is, the recovered images and the original images are exactly the same. And the PSNR values of the images recovered with ~~incorrect~~wrong keys are smaller ~~then~~than 20dB, which ~~display~~show that the recovery results are still very different from the original ones ~~de~~due to the noise surviving~~interference living~~ in the watermark areas of the healedrecovered images.

Figure 3.5. Extracts of stego-documents produced using the proposed method with databases 1 and 2.

# Chapter 4

# Quotation Authentication: A New Approach and Efficient Solutions by Data Hiding and Cascade Hashing Techniques

## 4.1 Introduction

The advancement of digital technology and the Internet has greatly eased the publication of information to the mass, whether in the form of web pages, e-mails, PDF files, or office documents. The abundance of contents made available is mostly beneficial to the general public, but significant amounts of annoying or even harmful information are also being distributed on the Internet.

There are many ways to help identify useful information. Search engines, such as Google and Yahoo, analyze the inter-linkage of documents on the web and deduce the relative degrees of importance of the documents by using their page ranking technology [88], with the result that documents ranked high in their search results tend to be more trustworthy. Also, people tend to trust information published by *credible sources*, for example, announcements by federal or local governments; publications from the IEEE, ACM, or other famous publishers with stringent peer review processes; news reported by CNN, ABC, or other well-known news agencies; and so on.

There are however numerous information publishers and distributors which are less known to the public and yet provide useful information. In particular, they sometimes organize valuable contents published by credible sources by quoting, aggregating, or disseminating them to make the edited data more relevant or accessible to readers. For example, it is common for an article to quote significant findings from technical journals or government reports that would otherwise never be read by the general public.

Unfortunately, there is no easy way for a reader to verify that the quotations contained in these documents are *really* from the claimed sources. This vulnerability is frequently used in *Internet hoaxes*, such as virus warnings and sympathy letters. One mobile phone virus hoax, for example, claims that if a user answers a call without caller identity, the user's phone can be infected and no longer usable. The hoax lures the reader by claiming that the information has been confirmed by credible mobile phone companies (like Nokia and Motorola), and that the news was reported on CNN. Due to the credibility of the claimed sources, most people forwarded the message without performing any further verification. The multiplicative effect of such blind message passing puts unnecessary burdens on the network and servers, and costs the reader valuable time to process them. Another example is nutrition and health recommendations that point out unhealthy food, products, or life styles. The contents are mixed with myths, true pieces of information, forged allegations on competitor companies, and so on. Such contents come from a variety of sources, some of which may not be accessible online or require special subscriptions, making it difficult for a reader to verify the truthfulness of the information.

In this study, we investigate this problem of *quotation authentication*, where a new approach is proposed to allow efficient authentication of quotations from trusted sources but embedded in documents created by unknown authors. The three parties involved in quotation authentication are respectively the *source author* $A^S$ of a source document $S$, the *document author* $A^D$ of a document $D$ that contains quotations from $S$, and the *document reader* $R^D$ of $D$. We assume that $R^D$ recognizes $A^S$ as a credible source but does not know $A^D$. The goal of quotation authentication is to allow $R^D$ to efficiently and undisputedly *verify* the fidelity and source of the quotations contained in $D$.

It is noted that methods to authenticate a message *as a whole* are relatively well studied. Typically, two communicating parties are assumed to share a secret key that is used to generate an appropriate message authentication code (MAC) for verifying the fidelity of contents transmitted over an insecure channel. Bellare et al. [90], [91] proposed the keyed-hash message authentication code (HMAC) that can generate provably secure message authentication codes based on standard hash functions such as MD5 and SHA1. Cipher-based MAC (CMAC) techniques such as XMAC and OMAC use standard block ciphers such as DES and AES to construct message authentication codes [92]. In the *asymmetric* case, a sender creates, using its *private key*, an appropriate *digital signature* for the transmitted content, such that the receiver can verify the fidelity of the content using the *public key* of the sender. A straightforward way to generate a digital signature is to generate a hash value for the content and encrypt the value using asymmetric encryption such as the RSA algorithm, as described in PKCS#1 [93]. A related research area is digital certificate works, which provide means for associating the human recognizable identity of an entity with that entity's public key. HTTPS is an example of such works where websites register and use X.509 certificates that are issued by trusted certificate authorities [94], allowing users to authenticate the websites they connect to.

We will describe in the subsequent sections how to efficiently generate quotation signatures using the above-mentioned primitives. Specifically, in the proposed method we assume that signatures contain (or contain a reference to) an appropriate digital certificate of the source author so that a document reader can obtain a source author's public key and verify its fidelity. The public key can then be used to authenticate quotations that were previously signed using the source author's private key.

There is a branch of research in stream authentication [95] that bears some resemblance to the problem of quotation authentication, where receivers need to verify whether the received stream of packets are from the intended source. The problem is made complex when dealing with authenticating multicast traffic in the face of packet losses. Efficient schemes for multicast authentication have been proposed by using the temporal relationships of network packets, that is, later packets can be used to verify the authenticity of prior packets [96]. These techniques, however, cannot be applied to the problem of quotation authentication studied here since there's no notion of time in documents and quotations.

The importance to verify quotations within a document was recognized by Fernstrom [89]. However, assumed in the proposed solutions is either that the document reader has access to the original source document or that there is a trusted party to which document authors can send arbitrary quotations and source documents in order to get endorsed quotations that are verifiable by the document reader.

## 4.2 Overview of Proposed Method

A more practical approach is proposed in this dissertation to solve the quotation authentication problem, where each of the three parties only needs to perform certain simple and deterministic processes, as described by the following scenario:

1. $A^S$ processes the source $S$ to create a certain *source signature* $G^S$, and then publishes $S$ along with $G^S$ to the general public;
2. $A^D$ cites a text segment $q$ of interest within $S$, uses $G^S$ to create an appropriate *quotation signature* $G^q$ for $q$, applies data hiding techniques to create an integrated *authenticable-quotation* $q'$, and puts $q'$ into $D$;
3. finally $R^D$, when reading $D$, verifies that the quotation $q$, which is contained in $q'$, is indeed authored by $A^S$.

The benefit of using data hiding techniques in Stage 2 above to create an integrated authenticable-quotation $q'$ is that the quotation can be distributed multiple times using standard copy-and-paste operations by several document authors without hindering the ability for the final document reader to identify such authenticable-quotations and to verify that the quotation is indeed authored by $A^S$. This is illustrated in Figure 4.1 below.

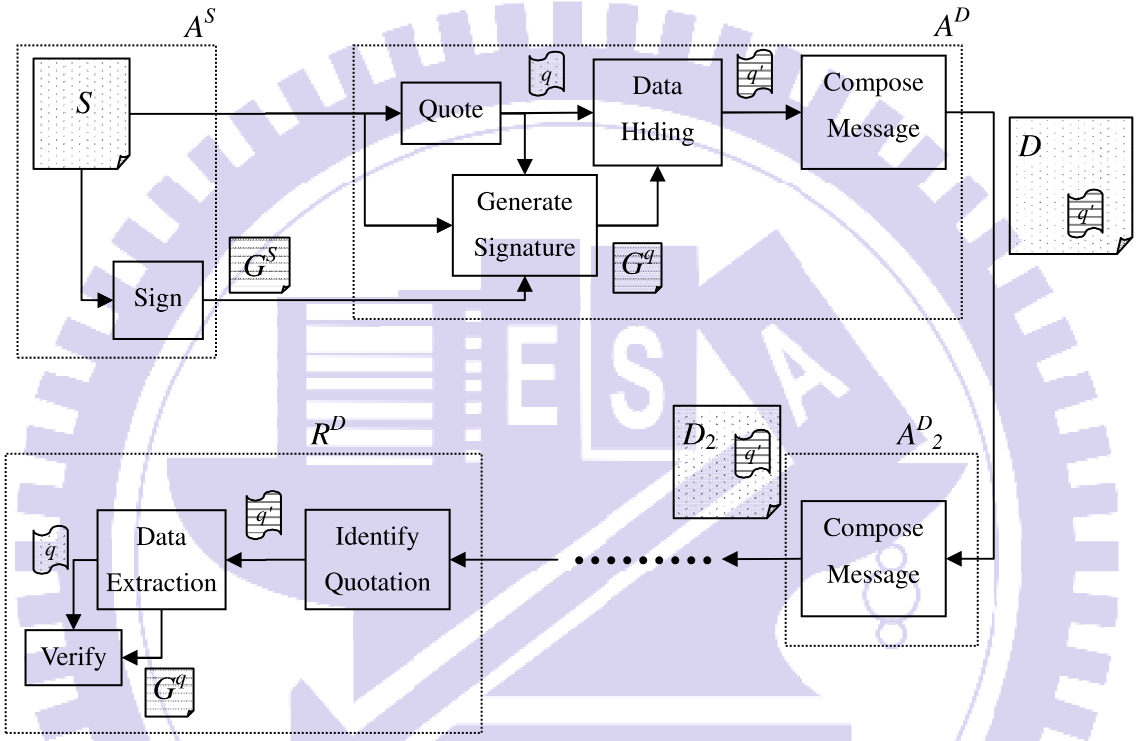


Figure 4.1. Illustration of processes performed by and information passed between a source author, one or more document authors, and a document reader.

It is assumed in the above proposed method that an appropriate quotation signature $G^q$ can be generated using the source signature $G^S$, thus alleviating the need for a third-party to endorse quotations. Also, $G^q$ alone is sufficient for $R^D$ to verify that the quotation $q$ is indeed from $A^S$, so that $R^D$ does *not* have to access the original source $S$. Although the digital signatures $G^S$ and $G^q$, attached to $S$ and $q$, respectively, are the overhead required to achieve the goal of quotation authentication, yet we propose techniques for generating appropriate source and quotation signatures to minimize the sizes of the overhead, as described in the sequel. In more detail, assume that texts are quoted from $S$ by $P$ document authors to generate $P$ documents, which are consumed by $Q$ document readers. The size of the total overhead of all $G^S$ and $G^q$ for such quotation authentication, called *total overhead size*, is $P|G^S| + P|G^q| +$

$Q|G^q|$, since each of the $P$ document authors needs to receive $S$ as well as $G^S$ from the source author and include $q$ as well as $G^q$ in his/her document, and each document reader needs to receive $q$ and $G^q$ from a document author.

## 4.3 Basic Signature Generation Techniques for Quotation Authentication

In this section, we describe two basic techniques to generate appropriate source and quotation signatures for the purpose of quotation authentication, and point out their shortcomings. The proposed more efficient techniques are then described subsequently.

A basic quotation authentication technique is for the source author to generate a digital signature for *every* possible quotation in a source document and to publish the signatures along with the source document on a certain website. A document author then only needs to quote the desired text and bundle it with the associated published digital signature to generate a valid authenticable-quotation. This technique is however inefficient since a quotation may start and end at any position of a document, and so the number of possible quotations required for a document of length $L$ is, as can be figured out, of the complexity order $O(L^2)$. This means in turn that the number of digital signatures is also of the order $O(L^2)$. This scheme is referred to as the *enumerate-all-quotations* technique in the sequel.

Another basic technique is to include the *entire* source document $S$, together with an appropriate digital signature, into the quotation signature $G^q$ of a quotation $q$. A document reader can then easily extract $S$ from $G^q$ and verify that the quotation really came from $S$. However, including the whole article will bloat the size of $G^q$ to be of the order $O(L)$, and the act of attaching the entire article $S$ might violate copyright law. This scheme is referred to as the *quote-the-whole* technique in the sequel.

The total overhead size of the enumerate-all-quotations technique is of $O(PL^2 + P + Q)$, or equivalently, $O(PL^2 + Q)$ since $G^S$ contains $O(L^2)$ digital signatures and $G^q$ contains a single digital signature. On the other hand, the total overhead size of the quote-the-whole technique is of $O(P + PL + QL)$, or equivalently, $O(PL + QL)$, since $G^S$ is a simple digital signature, but the size of $G^q$ is of the order $O(L)$.

If $Q$ is significantly larger than $P$, that is, if there are many more document readers than document authors, then the enumerate-all-quotations technique is more efficient than the quote-the-whole one in view of the total overhead size. However, if the magnitudes of $P$ and $Q$ are comparable, then the quote-the-whole technique is more efficient. Since the two techniques have their respective merits for different application cases, we propose in this study two better techniques which improve them respectively with the same goal of minimizing the total overhead size, as described in the following.

## 4.4 Multi-Use Signatures Technique (MUST)

The basic idea of the enumerate-all-quotations technique is to include in the source signature $G^S$ the digital signatures for all possible quotations so that the size of the quotation signature $G^q$ is small for any quotation. This is, however, overly inefficient for large $S$, and we describe in the following a *multi-use signatures technique* (MUST) where the size of $G^q$ remains to be of the order $O(1)$ for all quotations while the size of $G^S$ required is reduced to be of the order $O(L)$ instead of $O(L^2)$.

Assume that $S$ consists of $L$ *concatenated sentences* denoted as $s_1 \| s_2 \| s_3 \| \ldots \| s_L$ where "$\|$" specifies the concatenation operator, and that the length of each sentence is *bounded*, meaning that each sentence $s_i$ is of the order $O(1)$ in size. Also assumed is that the quotations include *complete* sentences, that is, $q$ includes a set of consecutive sentences in $S$ such that $q = s_a \| s_{a+1} \| \ldots \| s_b$ where $1 \leq a \leq b \leq L$. This assumption is reasonable because quoting only a portion of a sentence may change its meaning significantly. If this is not the case, a remedy technique has also been proposed in this study, as described later in Section 4.6.2.

### 4.4.1 Generation of MUST Source and Quotation Signatures

The idea of the proposed MUST is for the source author to generate $L$ *multi-use signatures* $g_j$, where $1 \leq j \leq L$, such that the signature $g_j$ can be used as part of the quotation signature for any of the quotations $s_a \| s_{a+1} \| \ldots \| s_j$, where $1 \leq a \leq j$. In addition to the multi-use signatures, we also generate $L$ *cascaded hash values* $h_j$, where $1 \leq j \leq L$, and include the hash value $h_j$ into the quotation signature for a quotation $s_j \| s_{j+1} \| \ldots \| s_b$, where $j \leq b \leq L$. The generation of the multi-use signatures and cascaded hash values is described in detail in the following algorithm.

**Algorithm 4.1: generation of a MUST source signature.**

***Input***: a source document $S$ consisting of $L$ sentences $s_j$ where $1 \leq j \leq L$.

***Output***: a source signature $G^S$, which contains $L$ cascaded hash values $h_j$ and $L$ multi-use signatures $g_j$ where $1 \leq j \leq L$.

***Steps***:

1. Set the first cascaded hash value to be the hash value of the entire source, that is, set $h_1 = \mathrm{H}(S)$, where $\mathrm{H}(\cdot)$ is some hash function.
2. For each $j$ from 2 to $L$, compute the cascaded hash value $h_j$ as $h_j = \mathrm{H}(h_{j-1} \,\|\, \mathrm{H}(s_{j-1}))$.
3. For $1 \leq j \leq L$, calculate the multi-use signature $g_j$ as $g_j = \mathrm{Sign}(h_j \,\|\, \mathrm{H}(s_j))$, where $\mathrm{Sign}(\cdot)$ is a signing function of some digital signature algorithm.

As a simple example, the cascaded hash values generated by the above algorithm for a source document with three sentences is as follows: $h_1 = \mathrm{H}(S)$, $h_2 = \mathrm{H}(h_1 \,\|\, \mathrm{H}(s_1))$, and $h_3 = \mathrm{H}(h_2 \,\|\, \mathrm{H}(s_2))$. And the quotation signature for the quotation $q = s_3$ contains $h_3$ and $g_3$, where $g_3 = \mathrm{Sign}(h_3 \,\|\, \mathrm{H}(s_3))$. Note that in detail $h_3$ equals $\mathrm{H}(h_2 \,\|\, \mathrm{H}(s_2)) = \mathrm{H}(\mathrm{H}(h_1 \,\|\, \mathrm{H}(s_1)) \,\|\, \mathrm{H}(s_2)) = \mathrm{H}(\mathrm{H}(\mathrm{H}(S) \,\|\, \mathrm{H}(s_1)) \,\|\, \mathrm{H}(s_2))$, which is a value derived by a cascade of concatenation and hashing operations, hence the name cascaded hash value.

### 4.4.2 Verification of MUST Quotation Signatures

When a document reader receives a quotation $q = s_a \,\|\, s_{a+1} \,\|\, \ldots \,\|\, s_b$ and a quotation signature $G^q$ that includes $h_a$ and $g_b$, the following algorithm is proposed to verify the quotation.

**Algorithm 4.2: quotation verification using MUST.**

***Input*:** a quotation $q = s_a \,\|\, s_{a+1} \,\|\, \ldots \,\|\, s_b$, where $a \leq b$, a cascaded hash value $h_a$, and a multi-use signature $g_b$.

***Output*:** result of quotation verification.

***Steps*:**

1. Perform the following steps to compute $h_b$ while $a$ is smaller than $b$:
   a. compute the cascaded hash value $h_{a+1}$ to be $\mathrm{H}(h_a \,\|\, \mathrm{H}(s_a))$;
   b. increment the value of $a$ by 1.
2. Verify the quotation by $\mathrm{Verify}(h_b \,\|\, \mathrm{H}(s_b), g_b)$ and output the result, where Verify is the reciprocal digital signature verification function of the Sign function used by the source author.

The above verification works for any quotation $q = s_a \,\|\, s_{a+1} \,\|\, \ldots \,\|\, s_b$, where $1 \leq a \leq b \leq L$, because the cascaded hash value $h_b$ calculated by Algorithm 4.2 using $h_a$ and $q$ is the same as that calculated in Algorithm 4.1. This is easily seen since the computations performed by Step 1a of Algorithm 4.2 are the same as those by Step 2 of Algorithm 4.1.

Since the cascaded hash value $h_b$ is constructed by using cascaded operations of hashing and concatenation, an attacker needs to find exact collisions of hash values in order to craft a forged quotation as well as a matching quotation signature that can ensure the same $h_b$ to be constructed using Algorithm 4.2. The proposed technique is thus attack-resilient if the functions H and Sign are chosen properly, such as using the hash function SHA1 or SHA2 and the signing function RSA with sufficiently long keys [92], [97].

### 4.4.3 Total Overhead Size of MUST

A MUST source signature $G^S$ for a source document $S$ with $L$ sentences always contain exactly $L$ cascaded hash values and $L$ multi-use signatures, and so the size of $G^S$ is of the order $O(L)$. A MUST quotation signature $G^q$ always contain one cascaded hash value and one multi-use signature, and thus the size of $G^q$ is of the order $O(1)$. The total overhead size of the MUST with $P$ document authors and $Q$ document readers is thus $O(PL + P + Q)$, or equivalently, $O(PL + Q)$, contrasted with the total overhead size of $O(PL^2 + Q)$ for the basic enumerate-all-quotations technique.

## 4.5 Tree Root Uni-Signature Technique (TRUST)

The previously described quote-the-whole technique requires the complete source document $S$ to be included in the $G^q$ of a certain $q$. This is overly inefficient for large $S$, and the proposed improving technique, called *tree root uni-signature technique* (TRUST), is described in this section. The TRUST uses a *tree-like* construction of hash values such that the size of $G^q$ can be reduced to be of the order $O(\log_2 L)$. *Only the root* of the tree of the hash values needs to be signed by the source author, thus maintaining the size of $G^S$ to be of $O(1)$.

### 4.5.1 Generation of TRUST Source Signatures

We assume as before that $S$ consists of $L$ sentences and that quotations include complete sentences. In the proposed TRUST, the source author generates a binary tree of hash values $h^i_j$ from $S$ by the following algorithm, where $i$ is the depth of the tree node and $j$ is the index of the nodes at depth $i$. The hash value of the tree root $h^r_1$, where $r = \lceil \log_2 L \rceil + 1$, is then signed with some digital signature algorithm to get the *tree root uni-signature* $g^r_1$ = Sign($h^r_1$). We note that this part of signature generation is similar to that of a traditional Merkle tree.

**Algorithm 4.3: generation of a TRUST tree of hash values.**

***Input***: a source document $S$ consisting of $L$ sentences $s_1, s_2, \ldots, s_L$.

***Output***: a tree of hash values $h^i_j$, where $1 \le i \le r$, and $j$ is the index of the nodes at depth $i$.

***Steps***:

1. Calculate a hash value for each sentence $s_j$ to get the lowest-level hash values, that is, set $h^1_j = \mathrm{H}(s_j)$ for $1 \le j \le L$, where $\mathrm{H}(\cdot)$ is some hash function.
2. Concatenate the hash values in pairs and compute the second-level hash values as $h^2_j = \mathrm{H}(h^1_{2j-1} \| h^1_{2j})$ for $1 \le j \le \lfloor L/2 \rfloor$ where $\lfloor \cdot \rfloor$ means the floor operation. If $L$ is odd, take the final hash value for $j = (L + 1)/2$ to be $h^2_j = h^1_{2j-1}$.
3. Repeat the above step to calculate the third-level hash values $h^3_j = \mathrm{H}(h^2_{2j-1} \| h^2_{2j})$ and so on, each time calculating a half (plus one if odd) of the number of the hash values computed in the previous step, until finally the hash value $h^r_1$ is generated for the tree root.

As an illustration, the TRUST tree of hash values generated by the above algorithm for $L = 5$ is shown in Table 4.1 below. The number of hash values at each level $i$ is denoted as $|h^i|$.

Table 4.1. TRUST tree of hash values for five sentences.

| $i$ | Hash values | | | | | $\|h^i\|$ |
|---|---|---|---|---|---|---|
| 1 | $h^1_1 = \mathrm{H}(s_1)$ | $h^1_2 = \mathrm{H}(s_2)$ | $h^1_3 = \mathrm{H}(s_3)$ | $h^1_4 = \mathrm{H}(s_4)$ | $h^1_5 = \mathrm{H}(s_5)$ | 5 |
| 2 | $h^2_1 = \mathrm{H}(h^1_1 \| h^1_2)$ | | $h^2_2 = \mathrm{H}(h^1_3 \| h^1_4)$ | | $h^2_3 = h^1_5$ | 3 |
| 3 | $h^3_1 = \mathrm{H}(h^2_1 \| h^2_2)$ | | | | $h^3_2 = h^2_3$ | 2 |
| 4 | $h^4_1 = \mathrm{H}(h^3_1 \| h^3_2)$ | | | | | 1 |

### 4.5.2 Generation of TRUST Quotation Signatures

For any quotation $q$ from $S$, a TRUST quotation signature contains the signature $g^r_1$ generated by the source author and a quotation-dependent set $H^q$, called the *complementary hash set*, of some of the hash values generated by Algorithm 4.3 above. In more detail, for a quotation $q = s_a \| s_{a+1} \| \ldots \| s_b$ where $1 \le a \le b \le L$, the complementary hash set is selected using the algorithm below.

**Algorithm 4.4: generation of a TRUST complementary hash set.**

***Input***: a source document $S = s_1 \| s_2 \| \ldots \| s_L$ and a quotation $q = s_a \| s_{a+1} \| \ldots \| s_b$.

***Output***: a complementary hash set $H^q$.

***Steps***:

1. Calculate the TRUST tree of hash values $h^i_j$ for $S$ using Algorithm 4.3.
2. Set the hash set $H^q$ to be the empty set initially, and set the value of $i$ to be 1.
3. If $a$ is even, then add the value $h^i_{a-1}$ to $H^q$.
4. If $b$ is odd and is smaller than $|h^i|$, then add the value $h^i_{b+1}$ to $H^q$.
5. Set $a$ to be $\lceil a/2 \rceil$ and $b$ to be $\lceil b/2 \rceil$, where $\lceil \cdot \rceil$ means the ceiling operation.
6. Increment the value of $i$ by 1, and go to Step 3 if $i$ is smaller than $r$.

As an example, if $S$ has five sentences and we wish to quote the second and third sentences, then according to Algorithm 4.4, the value of $i$ is set to 1 initially, and is increased to 2, then to 3, and finally to 4; the value of $a$ decreases from 2 to 1, and remains at 1; and the value of $b$ decreases from 3 to 2, and then to 1. And so the hash values $h^1_1$, $h^1_4$, and $h^3_2$ are added to $H^q$. It can be seen that the hash value of the tree root $h^r_1$ can be reconstructed as follows using only $H^q$ and $q$:

$$h^r_1 = \mathrm{H}(h^3_1 \,||\, h^3_2) = \mathrm{H}(\mathrm{H}(h^2_1 \,||\, h^2_2) \,||\, h^3_2)) = \mathrm{H}(\mathrm{H}(\mathrm{H}(h^1_1 \,||\, h^1_2)) \,||\, \mathrm{H}(h^1_3 \,||\, h^1_4)) \,||\, h^3_2)$$
$$= \mathrm{H}(\mathrm{H}(\mathrm{H}(h^1_1 \,||\, \mathrm{H}(s_2))) \,||\, \mathrm{H}(\mathrm{H}(s_3) \,||\, h^1_4)) \,||\, h^3_2).$$

This property is used for quotation verification, as described in the following.

### 4.5.3 Verification of TRUST Quotation Signatures

It is assumed that a document reader $R^D$ can identify a TRUST authenticable-quotation $q'$ in a document and can extract or reconstruct from $q'$ the quotation $q = s_a \,||\, s_{a+1} \,||\, \ldots \,||\, s_b$, the values $a$ and $b$, the signature $g^r_1$, and the complementary hash set $H^q$. Also, we assume that $R^D$ can retrieve hash values from $H^q$ in the same order as hash values were added to $H^q$ by the document author, and that the magnitudes $|h^i|$ can be determined correctly. To verify $q$, $R^D$ performs the steps described by the following algorithm to reconstruct the tree root hash value using $q$ and $H^q$, and finally verify the quotation using $g^r_1$.

**Algorithm 4.5: quotation verification using the TRUST.**

***Input***: a tree root uni-signature $g^r_1$, a quotation $q = s_a \,||\, s_{a+1} \,||\, \ldots \,||\, s_b$ where $a \le b$, and a complementary hash set $H^q$.

***Output***: result of quotation verification.

***Steps***:

1. Set the value of $i$ to be 1, and calculate the lowest-level hash values for the sentences in the quotation, that is, set $h^1_j = \mathrm{H}(s_j)$ for $a \leq j \leq b$.
2. If $a$ is even, then retrieve the next value from $H^q$, which is $h^i_{a-1}$.
3. If $b$ is odd and is smaller than $|h^i|$, then retrieve the next value from $H^q$, which is $h^i_{b+1}$.
4. Set $a$ to be $\lceil a/2 \rceil$ and $b$ to be $\lceil b/2 \rceil$.
5. Compute the next-level hash values $h^{i+1}_j = \mathrm{H}(h^i_{2j-1} \,\|\, h^i_{2j})$ for $a \leq j \leq b$.
6. Increment the value of $i$ by 1, and go to Step 3 if $i$ is smaller than $r$.
7. Verify the quotation by Verify($h^r_1$, $g^r_1$) where $h^r_1$ is the tree root hash value and output the result.

Why $H^q$ is called the complementary hash set should now be clear from the above algorithm — it fills in where needed such that the next-level hash values can be calculated from the sequence of hash values $h^i_a$, $h^i_{a+1}$, …, $h^i_b$. The process continues until the final tree root hash value $h^r_1$ is computed. Since $h^r_1$ is constructed by repeated applications of hashing followed by concatenation, it is difficult for an attacker to craft a forged quotation and a corresponding hash set that can ensure the same value of $h^r_1$ to be constructed in the same way as that in the case of a legitimate quotation.

### 4.5.4 Total Overhead Size of TRUST

A TRUST source signature always contains just a single digital signature $g^r_1$, and is thus of the order $O(1)$. On the other hand, a TRUST complementary hash set can have no more than $2r$ hash values, and hence the size of $H^q$, and also that of $G^q$, have an order of $O(\log_2 L)$. The total overhead size of the TRUST is thus $O(P + P\log_2 L + Q\log_2 L)$, or equivalently, $O(P\log_2 L + Q\log_2 L)$, contrasted with the total overhead size of $O(PL + QL)$ for the basic quote-the-whole technique.

## 4.6 Data Hiding Techniques for Quotation Authentication

In this section, the proposed data hiding techniques for the purpose of quotation authentication in Microsoft Word documents is described. For simplicity, this study assumes that quotations contain only text and all formatting of text are ignored.

### 4.6.1 Quotation Authentication Add-in

The Microsoft Word application allows *add-ins* [98] to be installed to customarily expand their capabilities. A prototype quotation authentication add-in has been implemented in this study using C$^{\#}$.NET and VSTO technologies[10], which installs several buttons on the toolbar of the Word application, as shown in Figure 4.2. In this way, the three parties involved in the problem of quotation authentication can perform their respective tasks as simple as a click of a button.

In more detail, a source author performs a one-time setup using "QA Setup" of the add-in to select the desired technique to use (MUST or TRUST) and a file that contains the author's digital certificate and private key. Then, the source author can generate a source signature for any source document by simply clicking on the "QA Sign" button of the add-in, which performs the following steps:

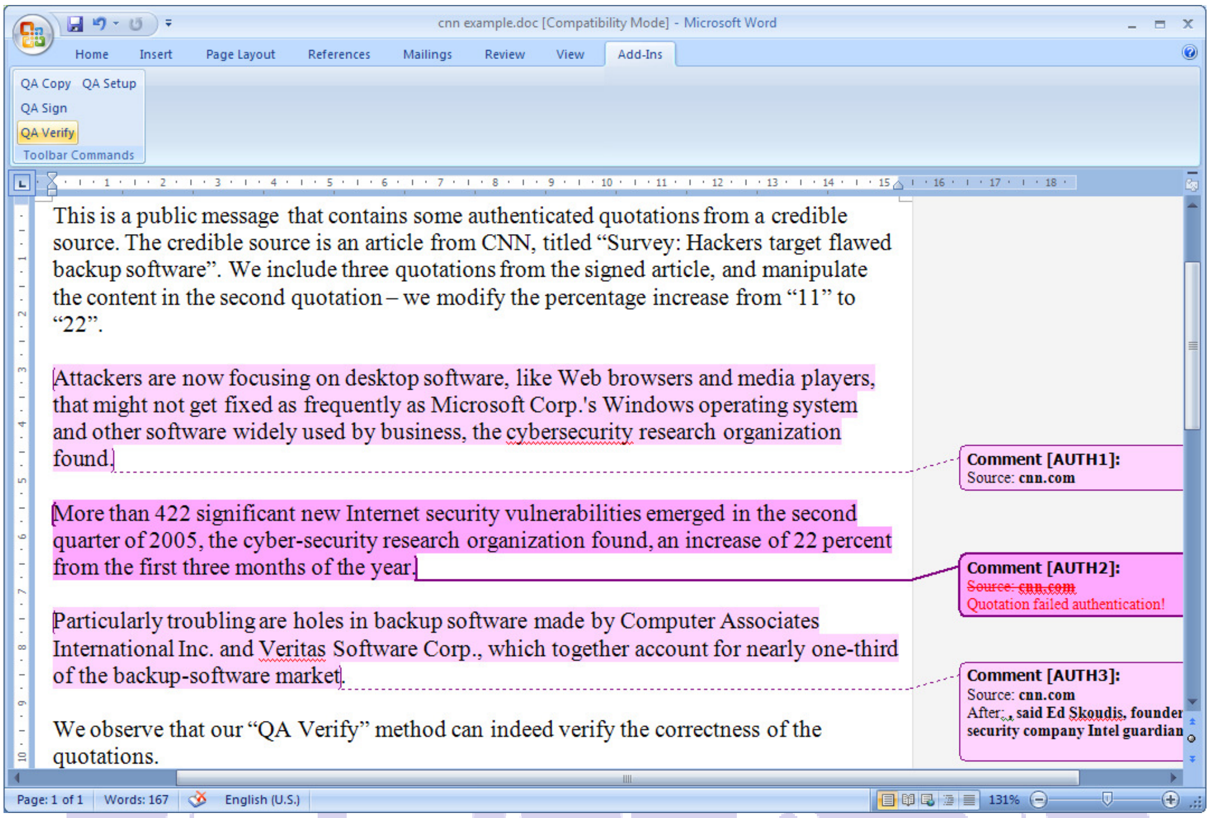
1. generate an appropriate source signature for the source document using the selected technique and the file containing digital certificate and private key information;
2. transform the binary source signature into text by Base64 encoding; and
3. store the transformed signature in the *document properties* of the document.

It is convenient to embed the source signature directly into the document as described above so that the signature does not have to be published separately. This is possible because Word documents allow arbitrary string values to be stored as key-value pairs in their special document properties dictionary.

When a document author receives a source document produced by the above process, he/she can select any quotation in the document and click on the "QA Copy" button of the add-in to activate the authenticable-quotation generation functionality, which performs the following steps:

1. extract the transformed text version of the source signature from the document properties dictionary of the source document;
2. transform the text signature back to the binary original using Base64 decoding;
3. generate a quotation signature $G^q$ for the selected quotation $q$ using the proposed technique as described previously;
4. create an integrated authenticable-quotation $q'$ from $q$ and $G^q$ using the proposed data hiding techniques described in the sequel; and put $q'$ into the system clipboard so that the document author can easily paste $q'$ into his/her own document.

[10] The implementation can be downloaded at http://sites.google.com/site/ktyliu/quotation-authentication-in-microsoft-word.



This is a public message that contains some authenticated quotations from a credible source. The credible source is an article from CNN, titled "Survey: Hackers target flawed backup software". We include three quotations from the signed article, and manipulate the content in the second quotation – we modify the percentage increase from "11" to "22".

Attackers are now focusing on desktop software, like Web browsers and media players, that might not get fixed as frequently as Microsoft Corp.'s Windows operating system and other software widely used by business, the cybersecurity research organization found.

More than 422 significant new Internet security vulnerabilities emerged in the second quarter of 2005, the cyber-security research organization found, an increase of 22 percent from the first three months of the year.

Particularly troubling are holes in backup software made by Computer Associates International Inc. and Veritas Software Corp., which together account for nearly one-third of the backup-software market.

We observe that our "QA Verify" method can indeed verify the correctness of the quotations.

**Comment [AUTH1]:**
Source: **cnn.com**

**Comment [AUTH2]:**
~~Source: cnn.com~~
Quotation failed authentication!

**Comment [AUTH3]:**
Source: **cnn.com**
After: **, said Ed Skoudis, founder**
**security company Intel guardian**

Figure 4.2. A screenshot of Microsoft Word with the prototype add-in installed, which has added buttons in the toolbar for the purpose of quotation authentication.

### 4.6.2 Integrated Authenticable-Quotation

An authenticable-quotation created by the prototype implementation carries three different pieces of information. The first is the quotation itself and the second is the human-readable source author identity, both of which should be conveyed to the document reader directly. The third is an appropriate quotation signature that can be processed by the add-in to verify the fidelity and source of the quotation, and should ideally be invisible to the reader.

We propose to store the source author information in the *visible comments* section of a Word document, as shown in Figure 4.2. Such comments are usually displayed at the right hand side in Microsoft Word and clearly recognizable. A comment in Microsoft Word can be attached to any range of texts in the document, and we use this property to demarcate the span of a quotation. Also, each comment in a Word document has an *author* field to help distinguish different commentators during collaborative editing. This author field is utilized in this study to mark comments that are used for quotation authentication by using a special value of "AUTH". Finally, if a document author quotes an incomplete sentence, we include the unquoted parts of the sentence in the comments as well for the reader’s reference, and also to allow reconstruction of complete sentences for the purpose of quotation authentication.

We propose to embed the authentication information invisibly into an authenticable-quotation by first transforming the binary data into normal text by Base64 encoding to avoid misinterpretation. The transformed text is then inserted just after the first character of the quotation and made invisible by setting its “Font Effects” to “Hidden” [99]. It has been verified by experiments that this technique can be used to embed arbitrary long information invisibly into a Microsoft Word document.

The integrated authenticable-quotation that contains the quotation itself, the visible comment, and the invisible authentication information as described is generated automatically by the add-in by a click of the button. It has been verified in experiments that the authenticable-quotation is always copied in its entirety in copy-and-paste operations, making it easy for a document author to include such authenticable-quotations when composing documents.

On receiving a document containing authenticable-quotations, a document reader can verify the quotations by clicking on the “QA Verify” button installed by the authentication add-in, which scans the document for comments that contain the special author field ‘AUTH’ and performs the following verification for each authenticable-quotation:

1. identify the span of a quotation by the range covered by the comment;
2. extract the hidden authentication information in the quotation and reversely transform the text version back to the binary original by Base64 decoding;
3. if there exists partial sentences in the comment then extract them and reconstruct the full quotation sentences;
4. verify the fidelity of the quotation using the proposed quotation verification technique; and format the comment of the quotation to show the result of quotation verification, as illustrated in Figure 4.2.

## 4.7 Summary

The problem of quotation authentication is described in this chapter, and a new approach to solving the problem has been proposed that allows document readers to efficiently verify quotations cited from known sources but embedded in messages by untrusted document authors. The proposed approach only requires the three parties involved in the problem to perform simple steps, without requiring a trusted third party to endorse quotations or requiring a document reader to access the original source document. Specifically, a source author is allowed to generate an appropriate source signature, such that any document author can generate a suitable quotation signature for arbitrary quotations from the source. The quotation signature is bundled together with the quotation using the proposed data hiding techniques to form an integrated authenticable-quotation that can easily be copied and pasted to any document. Finally, a document reader can identify any authenticable-quotations present in a document, and efficiently verify the source and the fidelity of the quotation.

We have started by describing the basic enumerate-all-quotations and quote-the-whole techniques, followed by the multi-use signatures technique and the tree root uni-signature technique that allow more efficient generation of source and quotation signatures. The two techniques have their respective merits depending on whether the message is widely distributed or not. The MUST is more efficient if there are a large number of document readers for a document, while the TRUST is better otherwise. The total overhead sizes and the signature sizes of the proposed techniques are summarized in Table 4.2 below.

Also, specific data hiding techniques suitable for embedding source and quotation signatures in Microsoft Word documents have been proposed to demonstrate the feasibility of the proposed techniques. Furthermore, add-ins that can be installed in the Microsoft Word applications were described, which allows the three parties of the quotation authentication problem to perform their tasks easily.

Table 4.2. Summary of total overhead sizes and signature sizes of the proposed techniques.

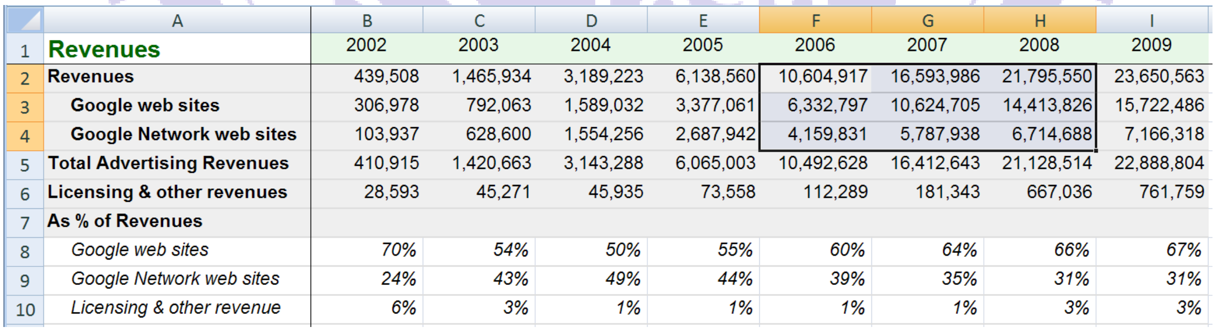
| Technique | $\lvert G^S \rvert$ | $\lvert G^q \rvert$ | Total overhead size |
|---|---|---|---|
| Enumerate-all-quotations | $O(L^2)$ | $O(1)$ | $O(PL^2 + Q)$ |
| Quote-the-whole | $O(1)$ | $O(L)$ | $O(PL + QL)$ |
| MUST | $O(L)$ | $O(1)$ | $O(PL + Q)$ |
| TRUST | $O(1)$ | $O(\log_2 L)$ | $O(P\log_2 L + Q\log_2 L)$ |

# Chapter 5
# Quotation Authentication and Content Authentication for Spreadsheet Documents

## 5.1 Introduction

The discussions in Chapter 4 assumed that documents and quotations contain texts that flow *consecutively*. However, contents in some documents are not sequential texts. For example, a Microsoft Excel spreadsheet document contains sheets of two-dimensional data, or *cells*. A typical quotation from a spreadsheet document is not sequential cells but a two-dimensional cutout of the cells. There are many large spreadsheets that contain information suitable for quoting, for example company financial statements, results of national voting, sales figures, and so on. It is common for a portion of such a large spreadsheet be quoted and included as a table in a Microsoft Word document.

As an example, a company's income statement may list the details of the revenue and expense figures in rows, with the values for different years or quarters across columns, as shown in Figure 5.1. If a business analyst (analogous to the document author in our problem of quotation authentication) quotes the figures of the revenues for a few selected years, the selection would be a two-dimensional subset of the spreadsheet, as illustrated in the figure.



| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | **Revenues** | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 |
| 2 | **Revenues** | 439,508 | 1,465,934 | 3,189,223 | 6,138,560 | 10,604,917 | 16,593,986 | 21,795,550 | 23,650,563 |
| 3 | **Google web sites** | 306,978 | 792,063 | 1,589,032 | 3,377,061 | 6,332,797 | 10,624,705 | 14,413,826 | 15,722,486 |
| 4 | **Google Network web sites** | 103,937 | 628,600 | 1,554,256 | 2,687,942 | 4,159,831 | 5,787,938 | 6,714,688 | 7,166,318 |
| 5 | **Total Advertising Revenues** | 410,915 | 1,420,663 | 3,143,288 | 6,065,003 | 10,492,628 | 16,412,643 | 21,128,514 | 22,888,804 |
| 6 | **Licensing & other revenues** | 28,593 | 45,271 | 45,935 | 73,558 | 112,289 | 181,343 | 667,036 | 761,759 |
| 7 | **As % of Revenues** | | | | | | | | |
| 8 | *Google web sites* | *70%* | *54%* | *50%* | *55%* | *60%* | *64%* | *66%* | *67%* |
| 9 | *Google Network web sites* | *24%* | *43%* | *49%* | *44%* | *39%* | *35%* | *31%* | *31%* |
| 10 | *Licensing & other revenue* | *6%* | *3%* | *1%* | *1%* | *1%* | *1%* | *3%* | *3%* |

Figure 5.1. Two-dimensional quotation in a spreadsheet document. (Source: Google Investor Relations)

Although Microsoft Excel documents can contain multiple worksheets where each is a two-dimensional array of cells, for simplicity we assume in this study that a source spreadsheet document $S$ consists simply of $X$ columns and $Y$ rows of cells[11]. A cell at column $x$ and row $y$ is denoted to be $s_{x,\,y}$ where $1 \le x \le X$ and $1 \le y \le Y$, and each cell can be empty or can contain a value such as a text string or a numerical value. In this study we assume each cell value has a *string representation* and that $s_{x,\,y}$ means the string representation of the cell whenever the context is clear. For example, if the top-left cell contains the value 53, this is denoted as $s_{1,\,1}$ = "53."

A *two-dimensional quotation* in this study, denoted as $q^{(a,\,b)}{}_{(c,\,d)}$, is a rectangular cut-out of the cells of size $A \times B$ that has a top-left cell $s_{a,\,b}$ and a bottom-right one $s_{c,\,d}$ where $1 \le a \le c \le X$, $1 \le b \le d \le Y$, $A = (c - a + 1)$, and $B = (d - b + 1)$.

The basic signature generation techniques described in Section 4.3 can be applied to the two-dimensional case here, but are inefficient. Specifically, for a spreadsheet containing $X$ columns by $Y$ rows of cells, the enumerate-all-quotations technique requires the source author to generate a signature for every possible rectangle that can be quoted. Since a rectangular quote can have a top-left corner starting at any position and a right-bottom corner ending at any position as well, the number of signatures generated may be figured out to be of the order $O((X \times Y)^2)$. On the other hand, all $X \times Y$ cells need to be included in a quotation signature for the quote-the-whole technique. The total overhead size for the two techniques are thus $O(P(XY)^2 + Q)$ and $O(PXY + QXY)$, respectively.

We describe below two better techniques, 2D-MUST and 2D-TRUST, that improves the total overhead size to be $O(PXY + Q\min(A, B))$ and $O(P\log_2 XY + Q\log_2 XY)$, respectively. Furthermore, we show that the proposed 2D-MUST can be applied to authenticate the contents of a spreadsheet document effectively. Specifically, source signatures can be generated and embedded into a Microsoft Excel document, such that modifications to the cell contents, transpositions of rows or columns, as well as additions and removals of rows or columns can be detected and changes highlighted.

## 5.2 Two-Dimensional Multi-Use Signatures Technique (2D-MUST)

The proposed improving technique, called the *two-dimensional multi-use signatures technique* (2D-MUST) generates a set of cascaded hash values and multi-use signatures of

[11] Microsoft Excel documents can contain contents other than cells, but we limit our discussion to cells and their authentication in this study.

size $O(XY)$ such that these can be used to cover all two-dimensional quotations $q^{(a,b)}{}_{(c,d)}$ with a top-left cell $s_{a,b}$ and a bottom-right one $s_{c,d}$ where $1 \le a \le c \le X$ and $1 \le b \le d \le Y$. The quotation signature however is no longer $O(1)$ but linear with the minimum of the number of rows or columns that the quotation spans, as described in detail in the following.

### 5.2.1 Generation of 2D-MUST Source Signatures

The first part of generating source signatures is similar to that of the one-dimensional case, where each row is considered to be an independent one-dimensional document consisting of sequential cells. Specifically, we perform the first two steps of Algorithm 4.1 for each row where the input is taken to be the row's content $S_y = s_{1,y} \parallel s_{2,y} \parallel \ldots \parallel s_{X,y}$ for row number $y$ ($1 \le y \le Y$) to yield the cascaded hash values $h_{1,y}, h_{2,y}, \ldots, h_{X,y}$. However, we do not simply sign these cascaded hash values as we did in Step 3 of Algorithm 4.1. This is because for a quotation with a top-left cell $s_{a,b}$ and a bottom-right one $s_{c,d}$:

1. each row in the quotation needs a separate digital signatures, meaning that a total of $(d - b + 1)$ digital signatures need to be included in $G^q$;
2. each digital signature can only verify the integrity of that row's content, so the digital signatures cannot be used to detect transpositions of complete rows.

In the second part of the proposed technique, a second series of cascaded hash values, denoted as $h'_{x,y}$ where $1 \le x \le X$ and $1 \le y \le Y$, are generated. To avoid ambiguity, we call the first series of cascaded hash values the *1D cascaded hash values*, while the second series the *2D cascaded hash values*. The reason of this naming will be made apparent in the following.

The 2D cascaded hash values are generated in a downward direction in a column, in contrast to the 1D cascaded hash values that was generated in a rightward horizontal direction. Also, in contrast to the 1D cascaded hash values that use the cell contents to generate subsequent cascaded hash values, the 2D cascaded hash values are calculated using the 1D cascaded hash values, as illustrated in Figure 5.2 below. That is, whereas $h_{x+1,y}$ is calculated as $\mathrm{H}(h_{x-1,y} \parallel \mathrm{H}(s_{x,y}))$, $h'_{x,y}$ is calculated as $\mathrm{H}(h'_{x,y-1} \parallel h_{x,y})$. The details of the proposed technique are described below as an algorithm.

**Algorithm 5.1: generation of a 2D-MUST source signature.**

***Input***: a source document $S$ consisting of $X$ columns by $Y$ rows of cells $s_{x,y}$ where $1 \le x \le X$ and $1 \le y \le Y$.

***Output***: $X \times Y$ 1D cascaded hash values $h_{x,y}$, $X \times Y$ 2D cascaded hash values $h'_{x,y}$, and $X \times Y$

multi-use signatures $g_{x,\,y}$ where $1 \le x \le X$ and $1 \le y \le Y$, which are included as part of a source signature $G^S$.

***Steps*:**

1. For each $y$ from 1 to $Y$, compute the 1D cascaded hash values for row $y$ as follows.
   a. Set $h_{1,\,y} = \mathrm{H}(S_y)$ where $\mathrm{H}(\cdot)$ is some hash function and $S_y$ is the content of the whole row, that is, $S_y = s_{1,\,y} \parallel s_{2,\,y} \parallel \ldots \parallel s_{X,\,y}$.
   b. For each $x$ from 2 to $X$, compute $h_{x,\,y}$ as $h_{x,\,y} = \mathrm{H}(h_{x-1,\,y} \parallel \mathrm{H}(s_{x-1,\,y}))$.
2. For each $x$ from 1 to $X$, compute the 2D cascaded hash values for column $x$ as follows.
   a. Set $h'_{x,\,1} = \mathrm{H}(S'_x)$ where $S'_x$ is the content of the whole column, that is, $S'_x = s_{x,\,1} \parallel s_{x,\,2} \parallel \ldots \parallel s_{x,\,Y}$.
   b. For each $y$ from 2 to $Y$, compute $h'_{x,\,y}$ as $h'_{x,\,y} = \mathrm{H}(h'_{x,\,y-1} \parallel h_{x,\,y-1})$.
3. For each $y$ from 1 to $Y$ and for each $x$ from 1 to $X$, compute the multi-use signature $g_{x,\,y}$ as $g_{x,\,y} = \mathrm{Sign}(\mathrm{H}(h'_{x,\,y} \parallel \mathrm{H}(h_{x,\,y} \parallel \mathrm{H}(s_{x,\,y}))))$, where $\mathrm{Sign}(\cdot)$ is a signing function of some digital signature algorithm.

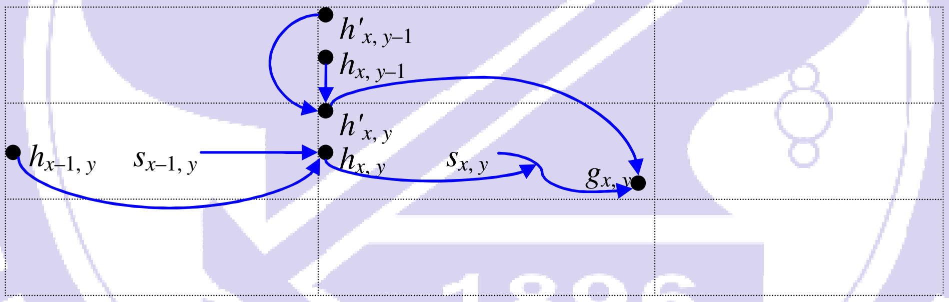


Figure 5.2. Illustration of cascaded hash value calculation for a cell $s_{x,\,y}$ in 2D-MUST.

### 5.2.2 Generation and Verification of 2D-MUST Quotation Signatures

When a quotation $q^{(a,\,b)}{}_{(c,\,d)}$ is quoted from $S$, an appropriate quotation signature $G^q$ is generated using $S$ and $G^S$ that includes the following.

1. A digital certificate of the source author $A^S$, so that the document reader can verify the association of the identity of $A^S$ and its public key.
2. The starting 1D cascaded hash values for each row in the quotation, that is, $h_{a,\,y}$ for $b \le y \le d$.
3. The starting 2D cascaded hash value for the last column in the quotation, that is, $h'_{c,\,b}$.
4. The multi-use signature $g_{c,\,d}$.

Similar to the case of one-dimensional MUST, the starting 1D cascaded hash values included in step 2 above are used by a document reader $R^D$ to regenerate the 1D cascaded hash values $h_{c,y}$ using $q^{(a,b)}{}_{(c,d)}$ for $b \le y \le d$ by a process similar to Step 1b of Algorithm 5.1. Then, the starting 2D cascaded hash value $h'_{c,b}$ and the regenerated 1D cascaded hash values $h_{c,y}$ where $b \le y \le d$ are used to calculate the 2D cascaded hash value $h'_{c,d}$ by a process similar to Step 2b of Algorithm 5.1. If any of the cells in the quotation has been modified, or if rows in the quotation has been transposed, then the value $h'_{c,d}$ so generated by $R^D$ will not be the same as that generated by $A^D$, and so failing the verification performed by $R^D$ – Verify(H($h'_{c,d}$ || H($h_{c,d}$ || H($s_{c,d}$)))), $g_{c,d}$) – where Verify($\cdot$) is the reciprocal digital signature verification function of the Sign($\cdot$) function used by the source author.

### 5.2.3 Total Overhead Size of 2D-MUST

A 2D-MUST source signature $G^S$ for a source document $S$ with $X$ columns by $Y$ rows of cells always contain exactly $X \times Y$ 1D cascaded hash values, $X \times Y$ 2D cascaded hash values, and $X \times Y$ multi-use signatures, and so the size of $G^S$ is of the order $O(XY)$. A 2D-MUST quotation signature $G^q$ for a quotation of size $A \times B$ contains $B$ 1D cascaded hash values, one 2D cascaded hash value, and one multi-use signature, and thus the size of $G^q$ is of the order $O(B)$. It can be figured out that the proposed technique may be flipped around, where 1D cascaded hash values are generated for columns and 2D cascaded hash values across rows, meaning that the quotation signature $G^q$ for a quotation of size $A \times B$ can be made to contain $A$ 1D cascaded hash values and hence to be of the size of order $O(A)$ instead of $O(B)$. If both types of cascaded hash values and multi-use signatures are generated by the source author, the size of $G^S$ will double but still be of the order $O(XY)$, while the size of $G^q$ will be $O(\min(A, B))$ since a document author can choose one of the two sets of cascaded hash values and multi-use signatures that yield the smaller quotation signature.

Consequently, the total overhead size of the 2D-MUST with $P$ document authors and $Q$ document readers, assuming that the average numbers of columns and rows quoted are $A$ and $B$ respectively, is $O(PXY + P\min(A, B) + Q\min(A, B))$, or equivalently, $O(PXY + Q\min(A, B))$, contrasted with the total overhead size of $O(P(XY)^2 + Q)$ for the enumerate-all-quotations technique.

## 5.3 Two-Dimensional Tree Root Uni-Signature Technique (2D-TRUST)

The next proposed improving technique, called the *two-dimensional tree root uni-signature technique* (2D-TRUST), uses a tree-like construction of hash values similar to that of the previously-described one-dimensional TRUST such that the size of quotation signatures can be reduced to be of the order $O(\log_2 XY)$, instead of $O(XY)$, for the basic quote-the-whole technique. Only the root of the two-dimensional tree of hash values needs to be signed by the source author, thus maintaining the size of the source signature to be of the order $O(1)$.

### 5.3.1 Generation of 2D-TRUST Source Signatures

For a source spreadsheet document $S$ consisting of $X \times Y$ cells, the source author generates a two-dimensional tree of hash values for the cells in $S$ in a way similar to that of the one-dimensional TRUST, as described in detail in the following algorithm. The hash value $h^r_{1,1}$ of the tree root is then signed with some digital signature algorithm to get a two-dimensional tree root uni-signature $g^r_{1,1} = \mathrm{Sign}(h^r_{1,1})$.

**Algorithm 5.2: generation of a 2D-TRUST tree of hash values.**

***Input*:** a source spreadsheet $S$ consisting of $X \times Y$ cells $s_{x,y}$ where $1 \le x \le X$ and $1 \le y \le Y$.

***Output*:** a two-dimensional tree of hash values $h^i_{x,y}$ where $i$ is the depth of the tree node; $x$ and $y$ are the indices of the nodes at depth $i$; and $h^r_{1,1}$ is the hash value of the tree root.

***Steps*:**

1. Calculate a hash value for each cell $s_{x,y}$ to get the lowest-level hash values, that is, set $h^1_{x,y} = \mathrm{H}(s_{x,y})$ for $1 \le x \le X$ and $1 \le y \le Y$.
2. Initialize the value of $i$ to be 1.
3. For all values of $x$ and $y$ where $1 \le x \le \lfloor X/2 \rfloor$ and $1 \le y \le \lfloor Y/2 \rfloor$, concatenate the hash values in fours to compute the next-level hash values as $h^{i+1}_{x,y} = \mathrm{H}(h^i_{2x-1,2y-1} \parallel h^i_{2x-1,2y} \parallel h^i_{2x,2y-1} \parallel h^i_{2x,2y})$.
4. Perform one of the following operations, depending on the values of $X$ and $Y$:
    a. if both $X$ and $Y$ are even, then set $X$ to be $X/2$ and set $Y$ to be $Y/2$;
    b. if $X$ is odd and $Y$ is even, then set $X$ to be $(X + 1)/2$ and $Y$ to be $Y/2$; and set $h^{i+1}_{X,Y} = \mathrm{H}(h^i_{2X-1,2Y-1} \parallel h^i_{2X-1,2Y})$;

c. if $X$ is even and $Y$ is odd, then set $X$ to be $X/2$ and $Y$ to be $(Y + 1)/2$; and set $h^{i+1}_{X,\,Y} = \mathrm{H}(h^{i}_{2X-1,\,2Y-1} \,||\, h^{i}_{2X,\,2Y-1})$;

d. if both $X$ and Y are odd, then set $X$ to be $(X + 1)/2$ and $Y$ to be $(Y + 1)/2$; and set $h^{i+1}_{X,\,Y} = h^{i}_{2X-1,\,2Y-1}$.

5. Increment the value of $i$ by 1.
6. Denote the numbers of columns and rows of hash values for level $i$ as $X^i = X$ and $Y^i = Y$, respectively.
7. Go to Step 3 if the tree root hash value has not been calculated, or equivalently, if $X$ or $Y$ is larger than 1.
8. Set the total number of levels $r$ to be $i$.

### 5.3.2 Generation of 2D-TRUST Quotation Signatures

When a document author $A^D$ quotes a two-dimensional rectangle of cells in a source spreadsheet, the quotation signature $G^q$ is generated by including the signature $g^r_{1,\,1}$ and a set of complementary hash values $H^q$ containing some of the hash values generated in Algorithm 5.2. Specifically, for a rectangular quotation $q^{(a,\,b)}_{(c,\,d)}$ with a top-left cell $s_{a,\,b}$ and a bottom-right one $s_{c,\,d}$ where $1 \le a \le c \le X$ and $1 \le b \le d \le Y$, the complementary hash set is selected using the algorithm below. The purpose of $H^q$ is the same as that in the one dimensional TRUST, that is, the tree root hash value $h^r_{1,\,1}$ can be reconstructed from $q$ and $H^q$.

**Algorithm 5.3: generation of a 2D-TRUST complementary hash set.**

***Input*:** a source spreadsheet $S$ consisting of $X\times Y$ cells $s_{x,\,y}$ where $1 \le x \le X$ and $1 \le y \le Y$, and a two-dimensional rectangle of quoted cells $q^{(a,\,b)}_{(c,\,d)}$ with a top-left cell $s_{a,\,b}$ and a bottom-right one $s_{c,\,d}$ where $1 \le a \le c \le X$ and $1 \le b \le d \le Y$.

***Output*:** a complementary hash set $H^q$.

***Steps*:**

1. Calculate from $S$ the tree of hash values $h^i_{x,\,y}$ using Algorithm 5.2, with $X^i\times Y^i$ hash values generated for level $i$.
2. Set $H^q$ to be the empty set initially, and set the value of $i$ to be 1.
3. Add to $H^q$ the following hash values so that the next-level hash values can be reconstructed from $q$:

a. if $a$ is even and $b$ is odd, then add the values $h^i_{a-1,\,b}$ and $h^i_{a-1,\,b+1}$ to $H^q$;

b. if $a$ is odd and $b$ is even, then add the values $h^i_{a,\,b-1}$ and $h^i_{a+1,\,b-1}$ to $H^q$;

c. if both $a$ and $b$ are even, then add the values $h^i_{a-1,\,b-1}$, $h^i_{a,\,b-1}$ and $h^i_{a-1,\,b}$ to $H^q$;

d. if $c$ is even, $d$ is odd, and $d < Y^i$, then add the values $h^i_{c-1,\,d+1}$ and $h^i_{c,\,d+1}$ to $H^q$;

e. if $c$ is odd, $d$ is even, and $c < X^i$, then add the values $h^i_{c+1,\,d-1}$ and $h^i_{c+1,\,d}$ to $H^q$;

f. if both $c$ and $d$ are odd, then:

i. add the value $h^i_{c+1,\,d}$ to $H^q$ if $c < X^i$;

ii. add the value $h^i_{c,\,d+1}$ to $H^q$ if $d < Y^i$;

iii. add the value $h^i_{c+1,\,d+1}$ to $H^q$ if $c < X^i$ and $d < Y^i$.

4. Set $a$ to be $\lceil a/2 \rceil$, $b$ to be $\lceil b/2 \rceil$, $c$ to be $\lceil c/2 \rceil$, and $d$ to be $\lceil d/2 \rceil$.
5. Increment the value of $i$ by 1, and go to Step 3 if $i$ is smaller than $r$.

As a simple example, when quoting a single cell $s_{3,\,2}$ from a 5×5 spreadsheet, we add the values $h^1_{3,\,1}$, $h^1_{4,\,1}$, and $h^1_{4,\,2}$ to $H^q$ when $i$ is 1 (note the hash value $h^1_{4,\,1}$ is added twice, once in Step 3b, and once in Step 3e). The values of $a$ and $c$ are then set to be 2 while the values of $b$ and $d$ are set to be 1 in Step 4. In the next iteration, the values $h^2_{1,\,1}$, $h^2_{1,\,2}$, and $h^2_{2,\,2}$ are added to $H^q$ when $i$ is 2, and the values of $a$, $b$, $c$, and $d$ are all set to be 1. In the last iteration, the hash values $h^3_{2,\,1}$, $h^3_{1,\,2}$, and $h^3_{2,\,2}$ are added to $H^q$ when $i$ is 3. The hash values added to $H^q$ for this example is illustrated in Table 5.1 below.

Table 5.1. Hash values selected in the 2D-TRUST complementary hash set when quoting a cell $s_{3,\,2}$ from a 5×5 spreadsheet.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | $h^2_{1,\,1}$ | | $h^2_{1,\,2}$ | | $h^3_{1,\,2}$ |
| 2 | | | | | |
| 3 | $h^1_{3,\,1}$ | $s_{3,\,2}$ | $h^2_{2,\,2}$ | | |
| 4 | $h^1_{4,\,1}$ | $h^1_{4,\,2}$ | | | |
| 5 | $h^3_{2,\,1}$ | | | | $h^3_{2,\,2}$ |

## 5.3.3 Verification of 2D-TRUST Quotation Signatures

It is assumed that a document reader $R^D$ can identify a 2D-TRUST authenticable-quotation in a message and can extract or reconstruct the two-dimensional quotation $q^{(a,\,b)}_{(c,\,d)}$, the signature $g^r_{1,\,1}$, the complementary hash set $H^q$, and the values $X^i$ and

$Y^i$ for $1 \le i \le r$. Also, we assume that $R^D$ can retrieve hash values from $H^q$ in the same order as hash values were added to $H^q$ by $A^D$. To verify $q$, $R^D$ performs the steps described by the following algorithm to reconstruct the tree root hash value $h^r_{1,1}$ using $q$ and $H^q$, and then verify the quotation using $g^r_{1,1}$.

**Algorithm 5.4: verification of a 2D-TRUST quotation signature.**

***Input*:** a two-dimensional quotation $q^{(a,b)}_{(c,d)}$ with a top-left cell of $s_{a,b}$ and a bottom-right one of $s_{c,d}$ where $a \le c$ and $b \le d$, a signature $g^r_{1,1}$, and a complementary hash set $H^q$.

***Output*:** result of quotation verification.

***Steps*:**

```
1. Set the value of $i$ to be 1, and calculate the lowest-level hash values for the cells in the
   quotation, that is, set $h^1_{x,y} = \mathrm{H}(s_{x,y})$ for $a \le x \le c$ and $b \le y \le d$.
2. Retrieve the following values from $H^q$ to calculate the next-level hash values:
   a. if $a$ is even and $b$ is odd, then retrieve the values $h^i_{a-1,b}$ and $h^i_{a-1,b+1}$ from $H^q$;
   b. if $a$ is odd and $b$ is even, then retrieve the values $h^i_{a,b-1}$ and $h^i_{a+1,b-1}$ from $H^q$;
   c. if both $a$ and $b$ are even, then retrieve the values $h^i_{a-1,b-1}$, $h^i_{a,b-1}$, and $h^i_{a-1,b}$;
   d. if $c$ is even, $d$ is odd, and $d < Y^i$, then retrieve the values $h^i_{c-1,d+1}$ and $h^i_{c,d+1}$;
   e. if $c$ is odd, $d$ is even, and $c < X^i$, then retrieve the values $h^i_{c+1,d-1}$ and $h^i_{c+1,d}$;
   f. if both $c$ and $d$ are odd, then:
      i.   retrieve the value $h^i_{c+1,d}$ from $H^q$ if $c < X^i$;
      ii.  retrieve the value $h^i_{c,d+1}$ from $H^q$ if $d < Y^i$;
      iii. retrieve the value $h^i_{c+1,d+1}$ from $H^q$ if $c < X^i$ and $d < Y^i$.
3. Set $a$ to be $\lceil a/2 \rceil$, $b$ to be $\lceil b/2 \rceil$, $c$ to be $\lceil c/2 \rceil$, and $d$ to be $\lceil d/2 \rceil$.
4. Compute the next-level hash values as $h^{i+1}_{x,y} = \mathrm{H}(h^i_{2x-1,2y-1} \| h^i_{2x-1,2y} \| h^i_{2x,2y-1} \| h^i_{2x,2y})$
   for $a \le x \le c$ and $b \le y \le d$.
5. Increment the value of $i$ by 1, and go to Step 2 if $i$ is smaller than $r$.
6. Verify the quotation by Verify($h^r_{1,1}$, $g^r_{1,1}$) and output the result.
```

The above algorithm basically retrieves values from $H^q$ where needed such that the next-level hash values can be calculated from the currently-available hash values in a form of a rectangle with a top-left hash value $h^i_{a,b}$ and a bottom-right one $h^i_{c,d}$. The process continues until the tree root hash value $h^r_{1,1}$ is computed finally. Since $h^r_{1,1}$ is constructed by repeated applications of hashing followed by concatenation, it is difficult for an attacker to craft a forged quotation and a corresponding hash set that can ensure the same value $h^r_{1,1}$ to be constructed in the same way as that in the case of a legitimate quotation.

### 5.3.4 Total Overhead Size of 2D-TRUST

A 2D-TRUST source signature always contains just a single digital signature $g^r_{1,1}$, and its size is thus of the order $O(1)$. On the other hand, it can be figured out that a 2D-TRUST complementary hash set contains no more than $6r$ hash values, and hence the size of $H^q$, and also that of $G^q$, have an order of $O(\log_2 XY)$. The total overhead size of the 2D-TRUST is thus $O(P\log_2 XY + Q\log_2 XY)$, contrasted with the total overhead size of $O(PXY + QXY)$ for the basic quote-the-whole technique.

### 5.3.5 Including Column and Row Headers in Quotations

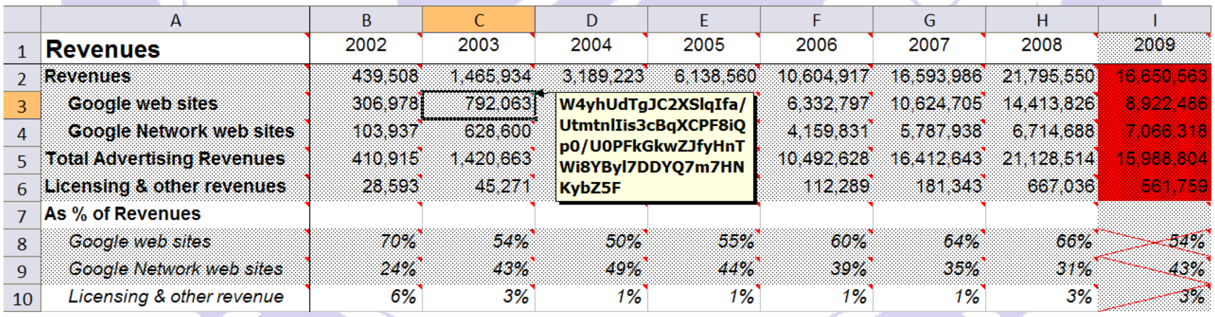
When a document author $A^D$ quotes a two-dimensional rectangle of cells, the corresponding column and row headers should also be included to put the quotation in context. In the example illustrated by Figure 5.1, the column headers specify the fiscal year of the quoted revenue figures (“2006”, “2007”, and “2008” in Figure 5.1), while the row headers specify the types of income or expenditure (“Revenues,” “Google web sites,” and “Google Network web sites” in Figure 5.1). Both headers are critical and thus must be quoted along with the cells containing the actual revenue figures.

Since column and row headers are typically one-dimensional, we can use the same techniques that are proposed in Section 4.4 or 4.5 to quote the corresponding headers efficiently. In cases where the column or row headers contain multiple rows/columns, we can apply the same technique (either 2D-TRUST or 2D-MUST) that is used for quoting cells to quote the extra column and row headers. It is noted that if the 2D-TRUST was used for both the quote as well as the headers, the size of the quotation signature required may actually *decrease* compared to only quoting the cells. This is because the headers can be used to calculate the first level hashes, and so reducing the number of hash values required for the complementary hash set for the quotation.

## 5.4 Authentication of Spreadsheet Contents

As mentioned previously, in addition to the application of quotation authentication, the proposed 2D-MUST can also be applied for effective authentication of the contents in a spreadsheet document. In more detail, a sender can generate and embed suitable cascaded hash values and multi-use signatures into a document and then send the stego-document to the receiver through an insecure channel. If an attacker modifies any of the cell's content during transit, the receiver can detect the modifications made.

In one of our experiments, we prototyped the 2D-MUST for authentication of a spreadsheet document by using C$^{\#}$.NET and VSTO technologies in Microsoft Excel 2003. Specifically, an add-in similar to that described in Section 4.6 was implemented[12], which added buttons to the toolbar of the Excel application for generating and verifying 2D-MUST signatures. The generated signatures for each cell, which include a 1D cascaded hash value, a 2D cascaded hash value, and a multi-use signature, are concatenated and stored into a cell as a *comment*. It is assumed that the sender and intended receiver of a document shares a private key and so the multi-use signature can be implemented using secure message authentication codes such as HMAC that has a significantly smaller size compared to asymmetric digital signatures. Our implementation used SHA1 as the hash function H(·) as well as for the signature generation and verification function (using HMAC backed by SHA1), and so the size of the signature for a cell is 3 × 160 bits. This signature is converted using base-64 encoding into an 80-character long string and included in the corresponding cell as a comment[13], as shown in Figure 5.3.



| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | **Revenues** | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 |
| 2 | Revenues | 439,508 | 1,465,934 | 3,189,223 | 6,138,560 | 10,604,917 | 16,593,986 | 21,795,550 | 16,650,563 |
| 3 | Google web sites | 306,978 | 792,063 | | | 6,332,797 | 10,624,705 | 14,413,826 | 8,922,486 |
| 4 | Google Network web sites | 103,937 | 628,600 | | | 4,159,831 | 5,787,938 | 6,714,688 | 7,066,318 |
| 5 | Total Advertising Revenues | 410,915 | 1,420,663 | | | 10,492,628 | 16,412,643 | 21,128,514 | 15,988,804 |
| 6 | Licensing & other revenues | 28,593 | 45,271 | | | 112,289 | 181,343 | 667,036 | 561,759 |
| 7 | As % of Revenues | | | | | | | | |
| 8 | *Google web sites* | *70%* | *54%* | *50%* | *55%* | *60%* | *64%* | *66%* | *54%* |
| 9 | *Google Network web sites* | *24%* | *43%* | *49%* | *44%* | *39%* | *35%* | *31%* | *43%* |
| 10 | *Licensing & other revenue* | *6%* | *3%* | *1%* | *1%* | *1%* | *1%* | *3%* | *3%* |

```
W4yhUdTgJC2XSlqIfa/
UtmtnlIis3cBqXCPF8iQ
p0/U0PFkGkwZJfyHnT
Wi8YByl7DDYQ7m7HN
KybZ5F
```

Figure 5.3. Experimental result of spreadsheet authentication using an add-in that implements the proposed 2D-MUST.

A receiver can easily verify the validity of a spreadsheet document by using the Add-In, which performs the following verification for each cell $s_{x,y}$ in the document.

[12] The implementation can be downloaded at http://sites.google.com/site/ktyliu/excel-spreadsheet-authentication.

[13] Microsoft Excel marks a cell that contains a comment by a small red triangle at the top-right corner of the cell and shows the content of the comment when the mouse is moved over a cell (the screenshot shown in Figure 5.3 was captured when the mouse was over the cell C2).

1. First, the comment of the cell is extracted and converted back into the three individual values $h_{x,y}$, $h'_{x,y}$, and $g_{x,y}$ using base-64 decoding.
2. The extracted signature $g_{x,y}$ is then compared against the result of computing Sign($h'_{x,y}$ || H($h_{x,y}$ || H($s_{x,y}$))). A mismatch between the two means the cell is likely to be modified and the Add-In will highlight the cell background color to be red.
3. For cells not in the first column, that is, $x > 1$, the extracted 1D cascaded hash value $h_{x,y}$ is compared against the result of computing H($h_{x-1,y}$ || H($s_{x-1,y}$)). A mismatch between the two means the cells $s_{x-1,y}$ and $s_{x,y}$ are likely not next to each other in the original spreadsheet document, and so we label the cell by drawing a diagonal line running from the top-left corner to the bottom-right one.
4. For cells not in the first row, that is, $y > 1$, the extracted 2D cascaded hash value $h'_{x,y}$ is compared against H($h'_{x,y-1}$ || $h_{x,y-1}$), a mismatch of which signals that the cells $s_{x,y-1}$ and $s_{x,y}$ are likely not next to each other in the original spreadsheet document. Cells failing the comparison are labeled with a diagonal line that slopes from the bottom-left corner to the top-right one in the prototype implementation.
5. Finally, the value $h_{1,y}$ is compared against H($S_y$) for each row $y$, and the value $h_{x,1}$ is compared against H($S'_x$) for each column $x$. A mismatch for each comparison likely means that either: 1) the row (or column) has been cropped; or 2) one or more cell contents in the row (or column) was modified; and so we highlight the whole row (column) by setting the cell background pattern to a dotted pattern.

In the example in Figure 5.3, the numbers in cells I2 thru I6 are individually decreased by some amount while the contents in cells I8 and I9 are copied from those in C8 and C9. It is verified that the modification to cells I2 thru I6 are correctly detected and highlighted (by step 2 of the above-mentioned verification process). Since the cells I8 and I9 are copied directly from C8 and C9, they passed the signature verification (step 2) but failed the cascaded hash value checks. Specifically, I8 failed both the 1D and 2D cascaded hash value checks (steps 3 and 4) while I9 failed the 1D cascaded hash value check (step 3). Although cell I10 was not modified, it failed the 2D cascaded hash value check (step 4), signaling that cells I9 and I10 are not sequential in the original spreadsheet document. Finally, column I as well as rows 2, 3, 4, 5, 6, 8, and 9 are highlighted with a background pattern by step 5, signaling that content in these columns/rows have been modified.

## 5.5 Summary

In this chapter we pointed out the existence of non-sequential quotations, in particular, quotations that are two-dimensional rectangles of cells in spreadsheet documents. The basic techniques of the enumerate-all-quotations and the quote-the-whole techniques may be applied for authentication of such quotations but are nevertheless inefficient in terms of the total overhead size. A two-dimensional multi-use signatures technique (2D-MUST) and a two-dimensional tree root uni-signature technique (2D-TRUST) were then proposed that can reduce the total overhead size, as summarized in Table 5.2 below.

Furthermore, it has been proposed that the 2D-MUST can be applied for effective authentication of spreadsheet contents. Experimental results were shown, demonstrating that various modifications to cells of a Microsoft Excel document can be detected and highlighted by using a prototype add-in that implements the proposed method.

Table 5.2. Summary of signature sizes and total overhead sizes of the proposed techniques.

| Technique | $\|G^S\|$ | $\|G^q\|$ | Total overhead size |
|---|---|---|---|
| Enumerate-all-quotations | $O((XY)^2)$ | $O(1)$ | $O(P(XY)^2 + Q)$ |
| Quote-the-whole | $O(1)$ | $O(XY)$ | $O(PXY + QXY)$ |
| 2D-MUST | $O(XY)$ | $O(\min(A, B))$ | $O(PXY + Q\min(A, B))$ |
| 2D-TRUST | $O(1)$ | $O(\log_2 XY)$ | $O(P\log_2 XY + Q\log_2 XY)$ |

# Chapter 6

# Invisible Watermarking in Slides of Presentations by Blank Space Coloring and Weighted Voting of Partial Sequences

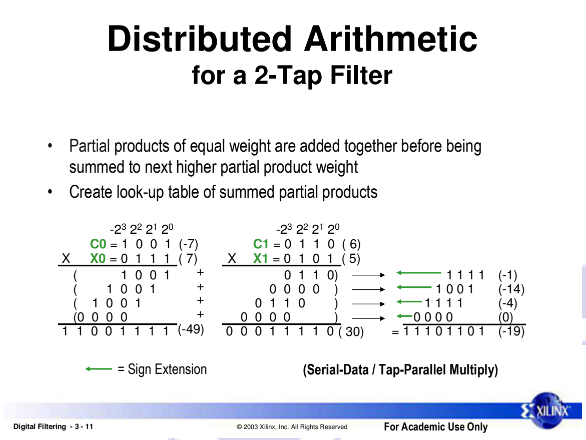
## 6.1 Introduction

Slide presentation is an increasingly popular way of communication, thanks to cheap projectors and a widespread deployment of them in institutions and businesses. Slide presentations are used for numerous purposes, including lecturing, training, idea presentation, and sales reporting. The slides of a presentation usually are crafted carefully and include texts, images, animations, audios, videos, etc., in order to present valuable contents concisely and lively.

It is common for a person to take slides from others' presentations when composing his/her own slide presentation. One should be careful before releasing such a composed presentation to the general public since including others' slides is an act of copyright violation. Another different but related application scenario is where there are confidential internal slides and public marketing slides in a company, and while it is perfectly fine to mix those slides in an internal talk, it is undesirable for the confidential slides to be carelessly shown in external presentations. It will be convenient if there is an automatic means to detect whether a set of slides contain any of the confidential slides.

One way to track the source of slides is to embed digital watermarks into them. To the best knowledge of the authors, digital watermarking of slides has not been investigated before. The traditional means to achieve source identification of slides is to place an annoying *visible* logo in the slide background.

In this chapter, a watermarking method for slide presentations is proposed, which embeds an *invisible* watermark image imperceptibly into the slides of a presentation. The embedded watermark survives common operations performed on the slides, such as copying and pasting of slides, addition of new slides, removal of slides, reordering of slides, editing of slide contents, and modification to the slide design. The last operation is often applied by a presentation designer to quickly change the style of slides for a desired appearance. The fonts, styles, and colors of texts in the slides, among others, are automatically modified according to a slide design template, as seen in the example shown in Figure 2.2 in the introductory chapter (the figure is repeated below in Figure 6.1 for convenience).



## Distributed Arithmetic
### for a 2-Tap Filter

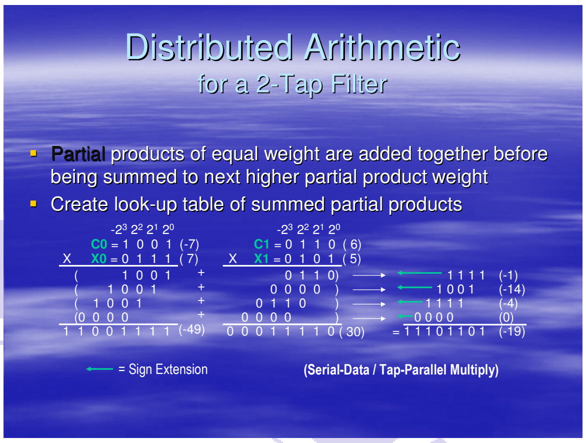
- Partial products of equal weight are added together before being summed to next higher partial product weight
- Create look-up table of summed partial products

```
      -2³ 2² 2¹ 2⁰              -2³ 2² 2¹ 2⁰
   C0 = 1  0  0  1 (-7)      C1 = 0  1  1  0 ( 6)
X  X0 = 0  1  1  1 ( 7)   X  X1 = 0  1  0  1 ( 5)
 (          1 0 0 1    +             0 1 1 0)  ──→  ←──── 1 1 1 1   (-1)
 (        1 0 0 1      +           0 0 0 0  )  ──→  ←───1 0 0 1     (-14)
 (      1 0 0 1        +         0 1 1 0    )  ──→  ←──1 1 1 1      (-4)
 (0 0 0 0              +       0 0 0 0      )  ──→  ←─0 0 0 0       (0)
 1 1 0 0 1 1 1 1 (-49)    0 0 0 1 1 1 1 0 ( 30)   = 1 1 1 0 1 1 0 1 (-19)
```

← = Sign Extension

**(Serial-Data / Tap-Parallel Multiply)**

Digital Filtering - 3 - 11 © 2003 Xilinx, Inc. All Rights Reserved **For Academic Use Only**



## Distributed Arithmetic
### for a 2-Tap Filter

- Partial products of equal weight are added together before being summed to next higher partial product weight
- Create look-up table of summed partial products

```
      -2³ 2² 2¹ 2⁰              -2³ 2² 2¹ 2⁰
   C0 = 1  0  0  1 (-7)      C1 = 0  1  1  0 ( 6)
X  X0 = 0  1  1  1 ( 7)   X  X1 = 0  1  0  1 ( 5)
 (          1 0 0 1    +             0 1 1 0)  ──→  ←──── 1 1 1 1   (-1)
 (        1 0 0 1      +           0 0 0 0  )  ──→  ←───1 0 0 1     (-14)
 (      1 0 0 1        +         0 1 1 0    )  ──→  ←──1 1 1 1      (-4)
 (0 0 0 0              +       0 0 0 0      )  ──→  ←─0 0 0 0       (0)
 1 1 0 0 1 1 1 1 (-49)    0 0 0 1 1 1 1 0 ( 30)   = 1 1 1 0 1 1 0 1 (-19)
```

← = Sign Extension

**(Serial-Data / Tap-Parallel Multiply)**

(a) (b)

Figure 6.1. Illustration of slide designs. (a) A slide from a tutorial from Xilinx, Inc. with black texts on white background; (b) the slide in (a) with a slide design template of bluish background applied.

There are two different ways of setting the colors of texts in presentations. The first, more common approach is to select a color from a color palette, and the selected *index* in the color palette is actually stored. The second approach is to directly set the color of the text usually in the RGB color space, which is a triplet specifying the relative intensities of the red, green, and blue components of the color, usually each in the range of 0 to 255, such as (*red*: 0, *green*: 255, *blue*: 255) for the color yellow. A slide editing software application usually allows any of the two approaches to be used for any text in a slide, and it is possible, for example, to have a whole sentence colored using the color palette approach except the first word which is highlighted using a special RGB color.

For automatic modifications of text colors to work when applying different slide design templates, the first approach of text coloring is used. In more details, different slide design templates have different color palettes, and the colors of the titles of a slide, the texts and the hyperlinks in a slide, etc. are set to specific colors from the color palette. The color palette in a design template supplied by a slide editing software application is designed to ensure high contrast texts and an overall appealing appearance. For example, slide templates with a white background will have matching black texts, and templates with a dark background will have texts in light colors. By adhering to the color palettes when editing slides, one can ensure that the colors will be modified appropriately and automatically when selecting a different slide design.

The automatic modifications made during slide design changes pose great challenges to watermarking in slides, and make many previously proposed watermarking techniques ineffective. As mentioned previously, text watermarking by making use of the LSBs of the text colors is revealed after a slide design change. This is because the manipulation of the LSBs of the text colors mean that the colors will no longer be palette entries but specific RGB colors, and thus not altered automatically when a different design template is applied, explaining why the word “Partial” in the right slide of Figure 6.1 remains to be dark gray instead of changing to white like the other text. It is also noted that the visible logo and the copyright information in the slide background have been *removed automatically* after the application of the slide design.

## 6.2 Overview of Proposed Method

In this chapter, we propose to use the *colors* of *space characters* (called simply as *spaces* hereafter) to embed watermarks, that is, to embed watermark data by altering the color of a space between two words. The colors of the spaces in a slide can be changed without affecting the visual appearance of the slide, and these colors are *unchanged* during application of other slide designs, changing of slide layouts, reordering of slides, reordering of texts in slides, and conversion of file formats, as found in this study. As the spaces are *transparent*, we can manipulate the colors freely for the purpose of information embedding, using either the color palette or the RGB coloring approach, without any visual side-effects. The latter is chosen due to its greater embedding capacity.

Specifically, we divide a watermark image into blocks and encode the index and data of each block into a RGB color value which is then taken to replace the original color of a space, accomplishing the embedding of a watermark block’s information into a space character, as illustrated in Figure 6.2 below. The watermark image to be embedded is assumed to be an $X \times Y$ black-and-white image, such as a logo of an institution. The image is divided into $M$ blocks, each containing $N$ pixels, where $N = XY/M$. The $N$ pixel values of each block are concatenated in a raster scan order into a string, which we call a *block data string* in the sequel. The data string and the index of each block are encoded into an RGB color, with which the color of a text space in a slide is replaced.

Since one might copy only some of the watermarked slides of a presentation, we choose to embed the watermark *repeatedly* throughout the slides. The embedding of the block index along with the block data string means that the embedded data are *invariant* against insertion and reordering of slides or slide contents. For resilience to common editing operations of slides, the watermark blocks are embedded into the slides in a *random sequence* created by a pseudo-number generator with a user-specified key. To extract the embedded watermark, a weighted voting technique is proposed to extract the embedded watermark from a stego-presentation.

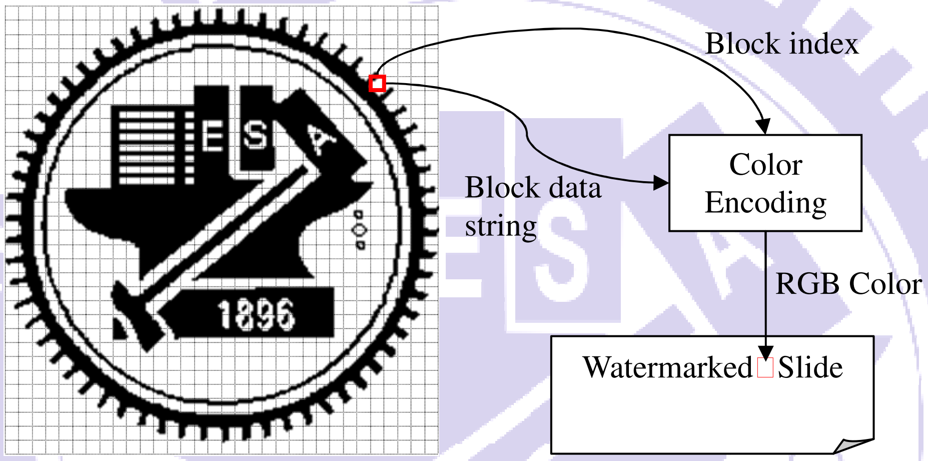


Figure 6.2. Illustration of watermark image embedding using blank space coloring.

The embedding of the blocks of a watermark image in a predefined random sequence has several benefits, as described in the following.

1. A recognizable partial watermark can be extracted if a person copies only a portion of the watermarked material. Figure 6.3 shows two series of watermark images with different percentages of blocks successfully reconstructed. The watermark can be recognized already when only half of the blocks (i.e., $M/2$ blocks) are present.
2. If some of the watermarked slides are placed together with other non-watermarked ones, the watermarked slides can still be correctly identified using the weighted voting technique proposed in this study (described later), which gives more weights to extracted block data strings with indices in right orders defined by the random sequence.
3. Furthermore, if watermarked slides from multiple sources with different user keys and watermark images are mixed together in a slide presentation, the individual watermark images can be extracted correctly in turn by using the respective user keys, as confirmed in the experiments.

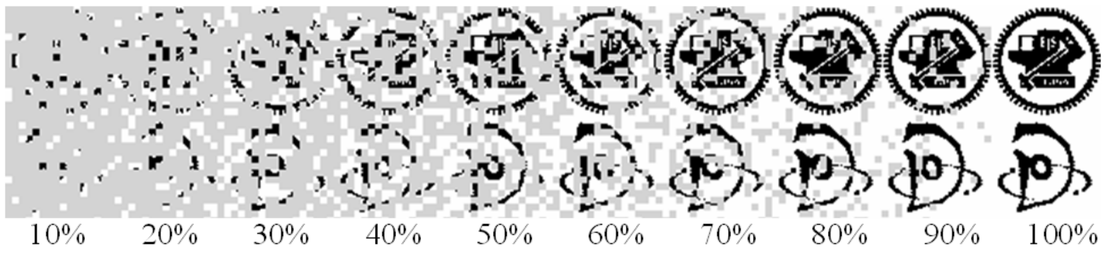


Figure 6.3. Two series of watermark logos with different percentages of blocks reconstructed.

## 6.3 Proposed Watermark Embedding Process

The detailed process of the proposed watermark embedding technique is described as an algorithm below. During watermark embedding, the spaces are taken for data embedding *in the reading/presentation order*, that is, the spaces in the first slide are used first in a top-to-bottom and left-to-right order, followed by the spaces in the second slide, and so on. While the blocks of the watermark are embedded into this normal sequence of spaces, the indices of the embedded blocks instead follow a pseudo-random sequence controlled by a key, as mentioned previously.

**Algorithm 6.1: embedding a watermark image into slides of a presentation.**

***Input*:** a set $P$ of slides of a presentation; a watermark image $I$ to be embedded, which is partitioned into $M$ block data strings $B_1, B_2, \ldots, B_M$; and a user-specified key $K$.

***Output*:** watermarked slides of $P$ with $I$ embedded by coloring the spaces in $P$ appropriately.

***Steps:***

1. Generate a random integer sequence $E = \{i_1, i_2, \ldots, i_M\}$ in the range of $\{1, 2, \ldots, M\}$ without repetitive values, using $K$ and a pseudo-random number generator $f$.
2. Find all spaces $s_1, s_2, \ldots, s_L$ in $P$ in the reading/presentation order, and repeat the sequence $E$ for $\lceil L/M \rceil$ times to arrive at another sequence $E^+ = \{j_1, j_2, \ldots, j_{L'}\}$, where $L' = M \times \lceil L/M \rceil \geq L$.
3. For each space $s_k$ in $P$, $1 \leq k \leq L$, pick out the index $j_k$ in $E^+$ and the corresponding block data string $B_{j_k}$, and encode the pair $(j_k, B_{j_k})$ into a color $C$ to replace that of $s_k$ in the following way:
   a. combine $j_k$ and $B_{j_k}$ into an integer $A = j_k \times 2^N + B_{j_k}$, regarding $B_{j_k}$ as an $N$-bit number;

b. compute color $C = (R, G, B)$ by taking the three components respectively to be $B = A_{\mathrm{mod}\ 2^n}$, $G = \lfloor A/2^n \rfloor_{\mathrm{mod}\ 2^n}$, and $R = \lfloor A/2^{2n} \rfloor$, where each component is assumed to have $n$ bits;

c. replace the color of space $s_k$ with $C$.

## 6.4 Proposed Weighted Voting of Partial Sequences Technique for Watermark Extraction

During watermark extraction, it might happen that a given set of slides includes both watermarked slides and non-watermarked ones, as mentioned previously. Since a space with no embedded data in a non-watermarked slide also has a color value which may be as well decoded into a block data string and a block index, a weighted voting technique is proposed in this study to identify the spaces that really contain watermark data, so that a correct watermark can be reconstructed. We assume that the block indices extracted from non-watermarked slides to be uniformly distributed in the range of $\{1, 2, \ldots, M\}$ for the subsequent discussions[14].

More specifically, since one usually copies a complete slide or an entire sentence in a slide at a time, the order of the spaces in the copied contents are preserved. The basic idea of the proposed weighted voting technique is to analyze the sequence of block indices extracted from the spaces of the slides of a suspect presentation, and check whether the extracted sequence follows an expected sequence. Blocks that follow the expected sequence are more likely to contain watermark data than those that do not, and thus are given larger weights in using them for reconstructing the watermark image.

We denote the sequence of pairs of block indices and block data strings that have been extracted from the spaces of a suspect presentation as $S = \{(j_1, B_1), (j_2, B_2), \ldots, (j_L, B_L)\}$, where $j_k$ is the index of block data string $B_k$. The expected sequence $E = \{i_1, i_2, \ldots, i_M\}$, as generated by Step 1 of Algorithm 6.1, is a random integer sequence in the range of $\{1, 2, \ldots, M\}$ without repetitive values. As the blocks of the watermark are embedded repeatedly according to the sequence $E$, we regard the sequence to be *cyclic* and use the new notation $E^+$ to specify a sequence of arbitrary length formed by concatenating sequence $E$ repeatedly.

[14] In practice, space characters in a typical slide presentation have very few distinct colors, so the performance of the proposed technique in reality is better than that of the theoretical calculations.

Also, we use the notation $\{j_a, j_{a+1}, \ldots, j_b\} \subset E^+$ to mean that the sequence $\{j_a, j_{a+1}, \ldots, j_b\}$ is a subsequence of $E^+$.

Now consider a text space $s_k$ which *does not* contain previously embedded watermark data, and from which the pair $(j_k, B_k)$ in $S$ is extracted. There are three cases here.

1. For $k = 1$ (corresponding to the case that $s_k$ is the first space of the suspect presentation), the probability that the index sequence $\{j_k, j_{k+1}\} \subset E^+$ is $1/2^M$, irrespective of whether the block with index $j_{k+1}$ contains embedded watermark data or not. The probability that $\{j_k, j_{k+1}\} \not\subset E^+$ is $1 - 1/2^M$.
2. For $k = L$ (corresponding to the case that $s_k$ is the last space of the suspect presentation), the probability that $\{j_{k-1}, j_k\} \subset E^+$ is $1/2^M$, irrespective of whether the block with index $j_{k-1}$ contains embedded watermark data or not. The probability that $\{j_{k-1}, j_k\} \not\subset E^+$ is $1 - 1/2^M$.
3. For $1 < k < L$, the probability that $\{j_{k-1}, j_k, j_{k+1}\} \subset E^+$ is $1/2^M \times 1/2^M = 1/2^{2M}$, irrespective of whether the blocks with block indices $j_{k-1}$ and $j_{k+1}$ contain embedded watermark data or not. The probability that $\{j_{k-1}, j_k, j_{k+1}\} \not\subset E^+$ but $\{j_{k-1}, j_k\} \subset E^+$ is $1/2^M \times (1 - 1/2^M)$, and so is the probability that $\{j_{k-1}, j_k, j_{k+1}\} \not\subset E^+$ but $\{j_k, j_{k+1}\} \subset E^+$. The probability that $\{j_{k-1}, j_k\} \not\subset E^+$ and $\{j_k, j_{k+1}\} \not\subset E^+$ is $(1 - 1/2^M) \times (1 - 1/2^M)$.

We propose to weigh the block data strings in $S$ according to the minus base-2 logarithm values of the above probabilities in using the data string values in the process of watermark extraction. That is, for $k = 1$, block data string $B_k$ is given a weight of $-\log_2(1/2^M)$, which we denote as $W_A$, if $\{j_k, j_{k+1}\} \subset E^+$; and if $\{j_k, j_{k+1}\} \not\subset E^+$, then block data string $B_k$ is given a weight of $-\log_2(1 - 1/2^M)$, which we denote as $W_B$. Similarly, if $k = L$, block data string $B_k$ receives a weight of $W_A$ if $\{j_{k-1}, j_k\} \subset E^+$, and a weight of $W_B$ if $\{j_{k-1}, j_k\} \not\subset E^+$. For $1 < k < L$, the block data string $B_k$ is given a weight of $-\log_2(1/2^{2M}) = 2W_A$ if $\{j_{k-1}, j_k, j_{k+1}\} \subset E^+$, a weight of $-\log_2[1/2^M \times (1 - 1/2^M)] = W_A + W_B$ if either $\{j_{k-1}, j_k\} \subset E^+$ or $\{j_k, j_{k+1}\} \subset E^+$, and a weight of $-\log_2[(1 - 1/2^M) \times (1 - 1/2^M)] = 2W_B$ if none of the above are true. The block data strings with the largest weights are then chosen for watermark reconstruction, as described in the following algorithm.

**Algorithm 6.2: weighted voting of partial index sequences for watermark extraction.**

***Input*:** an expected cyclic sequence $E^+ = \{i_1, i_2, \ldots\}$; an input sequence of extracted block indices and data strings $S = \{(j_1, B'_1), (j_2, B'_2), \ldots, (j_L, B'_L)\}$; and a threshold $T$.

***Output*:** block data strings $B_1, B_2, \ldots, B_M$ of the $M$ blocks that comprise a watermark image $I$.

***Steps:***

1. Initialize $W_1, W_2, \ldots, W_M$ to be $M$ empty sequences of 2-tuples.
2. For each pair $(j_k, B'_k)$ in $S$ with block index $j_k$ and block data string $B'_k$, $1 \leq k \leq L$, add the pair $(B'_k, W)$ to the sequence $W_{j_k}$ where:
   a. for $k = 1$, $W = W_A$ if $\{j_k, j_{k+1}\} \subset E^+$, and $W = W_B$ if $\{j_k, j_{k+1}\} \not\subset E^+$;
   b. for $k = L$, $W = W_A$ if $\{j_{k-1}, j_k\} \subset E^+$, and $W = W_B$ if $\{j_{k-1}, j_k\} \not\subset E^+$;
   c. for $1 < k < L$, $W = 2W_A$ if $\{j_{k-1}, j_k, j_{k+1}\} \subset E^+$; else $W = W_A + W_B$ if either $\{j_{k-1}, j_k\} \subset E^+$ or $\{j_k, j_{k+1}\} \subset E^+$; or else $W = 2W_B$.
3. Derive the data string $B_i$ of the $i$th block of $I$ as follows where $1 \leq i \leq M$:
   a. sum up the weights with identical data strings in $W_i$;
   b. select the data string $B'_{\max}$ in $W_i$ with the maximum weight $W_{\max}$;
   c. if there are no such $B'_{\max}$'s, or if there were multiple such $B'_{\max}$'s, or if $W_{\max}$ is smaller than the threshold $T$, then regard $B_i$ as *missing*, and represent the $i$th block as a *gray-colored block*; else set the data string $B_i$ to be $B'_{\max}$.

As an example, let $M = 4$, $E = \{3, 1, 4, 2\}$, $S = \{(2, \text{A}), (3, \text{B}), (1, \text{C}), (4, \text{D}), (1, \text{E}), (4, \text{F}), (2, \text{A})\}$, and $T = W_A$. Then $E^+ = \{3, 1, 4, 2, 3, 1, 4, 2, 3, \ldots\}$. After Step 2 of the above algorithm, we have $W_1 = \{(\text{C}, 2W_A), (\text{E}, W_A + W_B)\}$ because the partial data set $\{(3, \text{B}), (1, \text{C}), (4, \text{D})\}$ in $S$ forms an index sequence of $\{3, 1, 4\}$ which fits well with the first three indices of $E^+$, yielding the result of the pair $(\text{C}, 2W_A)$ in $W_1$; and the partial data set $\{(1, \text{E}), (4, \text{F})\}$ forms an index sequence of $\{1, 4\}$ found also in $E^+$, yielding $(\text{E}, W_A + W_B)$ in $W_1$. In similar ways, we can compute $W_2 = \{(\text{A}, W_A), (\text{A}, W_A)\}$, $W_3 = \{(\text{B}, 2W_A)\}$, and $W_4 = \{(\text{D}, W_A + W_B), (\text{F}, W_A)\}$. Accordingly, in Step 3 the block data strings $B_1$, $B_2$, $B_3$ and $B_4$ are set to C, A, B and D, respectively, with the weighting for data string A summed to be $2W_A$.

The above method ensures that sequences of blocks that contain watermark data dominate during the watermark image reconstruction in Step 3. However, if only a small portion of the watermarked contents are copied, then some of the blocks of the reconstructed watermark image may be *missing*. The threshold $T$ is useful in this case, where setting $T$ to a large value causes noise from non-watermarked blocks to be ignored. A value of at least $W_A$ is recommended, since block weights from any partial sequences of the watermark contents are at least of the value of $W_A$.

In Step 3c of the algorithm, instead of ignoring all of the multiple candidate block data strings with the same weights, we could use a voting algorithm to restore the correct watermark pixel values amid noise, as described in the following. The basic idea of the algorithm is that the block data strings decoded from text spaces that do not contain watermark data can be considered to contain random values. For each pixel, the number of blocks that have the corresponding pixel value of black is approximately the same as that having a pixel value of white. The assumption of inclusion of the correct block data string causes the scale to tip towards the correct side. To handle the case where no correct blocks is available, as the case may be when only a few watermarked slides are taken, the value of $H$ can be increased to reduce the resulting noise in the extracted watermark image.

**Algorithm 6.3: intra-block voting for pixel value reconstruction.**

***Input*:** a set $S_C$ of $V$ block data strings $B_1, B_2, \ldots, B_V$; and an adjustable threshold $H$, where $0.5 \leq H < 1$.

***Output*:** colors (black or white) $p_1, p_2, \ldots, p_L$ of the pixels comprising a block $P$ of a watermark image.

***Steps:***

1. Set the color of each pixel $p_j$, $1 \leq j \leq L$, as follows:
   a. count the number of blocks in $S_C$, whose corresponding pixel value is *black* (i.e., with bit value 1), and denote the number as $C_B$;
   b. count the number of blocks in $S_C$, whose corresponding pixel value is *white* (i.e., with bit value 0), and denote the number as $C_W$;
   c. set the color of $p_j$ to be black if $C_B / V > H$; else set the color of the pixel to be white if $C_W / V > H$; or else set the color of the pixel to be *gray*, meaning the pixel color was indeterminate.

## 6.5 Proposed Watermark Extraction Process

In the proposed watermark extraction process, the spaces in the slides of a suspect presentation are analyzed to extract a sequence of block indices and block data strings. Algorithm 6.2 is then used to analyze the extracted block indices and data strings to reconstruct the previously embedded watermark image. The algorithm below describes the details.

**Algorithm 6.4: extracting a watermark image from the slides of a suspect presentation.**

***Input*:** a set $P$ of slides of a suspect presentation and a key $K$.

***Output*:** a watermark image in $P$ comprised by $M$ block data strings $B_1, B_2, \ldots, B_M$.

***Steps:***

1. Generate the random integer sequence $E = \{i_1, i_2, \ldots, i_M\}$ in the range of $\{1, 2, \ldots, M\}$ without repetitive values, using $K$ and the same pseudo-random number generator $f$ used during watermark embedding.
2. Initialize $S$ to be an empty sequence of pairs of block indices and block data strings.
3. Find all spaces $s_1, s_2, \ldots, s_L$ in $P$ in the same order as that of embedding, and for each space $s_k$, $1 \leq k \leq L$, decode the color $C = (R, G, B)$ of $s_k$ into a pair $(j, D)$ of a block index $j$ and a block data string $D$ and put it into $S$ in the following way:
   a. compute an integer $A = R\times2^{2n} + G\times2^n + B$, assuming that the RGB color space has $n$ bits per channel;
   b. compute $j$ and $D$ as $j = \lfloor A/2^n \rfloor$ and $D = A_{\text{mod } 2^n}$, respectively (because presumably $A = j\times2^n + D$ according to Step 3a of Algorithm 6.1);
   c. add $(j, D)$ to $S$.
4. Reconstruct $B_1, B_2, \ldots, B_M$ using Algorithm 6.2 (and Algorithm 6.3) with $E$ and $S$ as inputs.

## 6.6 Embedding Capacity and Expected Reconstruction Coverage

When embedding the blocks of a watermark image into the spaces of slides, popular slide presentation formats like Microsoft PowerPoint and OpenOffice Impress can be used. Eight bits per color channel and hence 24 bits can be embedded into each text space in slides of such formats. This embedding capacity allows us to embed a black-and-white watermark image as large as 64×64 into the slides of a presentation of normal sizes. When embedding a watermark image of such a size, we first divide it into $M$ = 256 blocks with each block containing $N$ = 16 pixels. Each space then is used to store an 8-bit block index and 16-bits of pixel values. For a presentation we use in this study that contains slides with 40 spaces per slide on average, only seven slides is required to embed a complete watermark, and four slides may be sufficient to extract a recognizable watermark. The watermark image is embedded repeatedly into the slides as mentioned previously. Figure 6.3 shows a series of 64×64 logos with different coverages that have been divided into 256 blocks of 4×4 pixels. If a smaller watermark image was used during watermark embedding, the number of spaces required to extract a recognizable watermark is reduced.

When slides are taken selectively, instead of consecutively, from a watermarked presentation, the block data contained in these slides may overlap with each other, meaning that a higher number of spaces are required to reconstruct a recognizable watermark. We now analyze the estimated number of watermarked spaces that are required to achieve a particular watermark image *coverage* $G$, that is, the percentage of the watermark image that can be reconstructed. It is assumed that $R$ spaces are drawn from a watermarked presentation randomly, and that the block index in each of the drawn spaces is uniformly distributed in the sample space $\{1, 2, \ldots, M\}$. We denote the $R$ random block indices as $i_1, i_2, \ldots, i_R$, and so $G$ is essentially the number of *distinct values* in $\{i_1, i_2, \ldots, i_R\}$ over the value $M$. The *expected coverage* $E(G)$ is the expected percentage of the watermark image that can be reconstructed from the $R$ randomly chosen spaces.

To derive $E(G)$, we first introduce random variables $I_1, I_2, \ldots, I_M$, where $I_j = 1$ if none of the values of $i_1, i_2, \ldots, i_R$ is equal to $j$, and $I_j = 0$ if at least one of the values is equal to $j$. It should not be difficult to see that the probability of $I_j = 1$ is $(1 - 1/M)^R$, and thus the expected value of $I_j$ is $E(I_j) = (1 - 1/M)^R$. On the other hand, since $G = [\Sigma(1 - I_j)]/M$, the expected coverage can be derived to be

$$\begin{aligned} E(G) &= E\left(\frac{\sum(1-I_j)}{M}\right) \\ &= \frac{M-\sum E(I_j)}{M} \\ &= 1-\left(1-\tfrac{1}{M}\right)^R. \end{aligned} \tag{6.1}$$

From (6.1), the number of spaces $R$ required for a desired expected coverage $E(G)$ is

$$R = \frac{\log(1-E(G))}{\log\left(1-\tfrac{1}{M}\right)}. \tag{6.2}$$

Using the above equation, we can estimate the number of spaces that are required to achieve a recognizable watermark image. A recognizable watermark should have at least 50% coverage, as seen in Figure 6.4. With $E(G) = 0.5$ and $M = 256$, approximately $R = 177$ spaces, or about four to five slides, for a presentation containing on average 40 spaces per slide, are required according to Equation (6.2); and for a good quality watermark image with 80% coverage, approximately 411 spaces, or 10 slides, are required. On the other hand, if a smaller 20×16 watermark image was embedded, that is, $M = 16$ and $N = 20$, then only 11 spaces would be required to achieve 50% coverage and only 25 spaces to achieve 80% coverage. In other words, only about half a slide's worth of content is required for extracting a recognizable watermark of the size 20×16.

Finally, it is noted that in the case where a slide contained *few* spaces that can be used for watermark embedding using the proposed method, then the slide is likely to contain other multimedia contents such as images and drawings that they themselves can be used to embed watermarks by using the corresponding watermarking techniques for those media types.

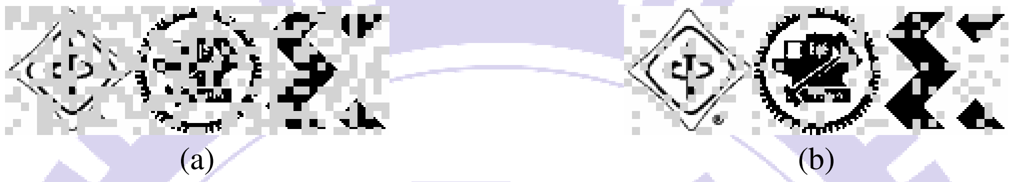

(a) (b)

Figure 6.4. Illustration of watermark reconstruction coverage. (a) Three watermarks each with a recovered percoverage of 50%; (b) the three watermarks with 80% coverage.

## 6.7 Robustness of Proposed Method against Common Operations

The watermark embedded into the slides of a presentation using the proposed method is resilient against many common operations performed on slides. In particular, the embedded watermark is robust against changes to the slide design template, as described in the introduction, whereas traditional visible logos are removed automatically in the process.

The watermark is resilient against copying and pasting of watermarked slides, as the colors and orderings of the spaces in the slides are unaltered during these types of operations. If we assume reasonably that there are at least two spaces in a slide, the block data strings embedded in the spaces of the slide will receive a sufficiently large weight using the proposed technique, allowing for correct reconstruction of the embedded watermark image. Reordering of the slides in a presentation is a similar operation to copying and pasting of slides, and has little impact on correct watermark extraction. Reordering of slide contents is often conducted by moving pictures and text blocks around or by exchanging the order of the sentences in a slide. The former operation does not have any effect, while the latter is the same as reordering of slides if there are at least two spaces in a sentence.

Insertion of new non-watermarked slides or watermarked slides created with a different key into a slide set does not affect the watermarked contents, but only increases the amount of noise during reconstruction. Algorithms 6.2 and 6.3 are capable of selecting out the correct watermark data amid noise, as is verified in the experiments conducted in this study, where correct watermark images were reconstructed from watermarked slides that have been reordered and put together with slides that have been watermarked with different keys.

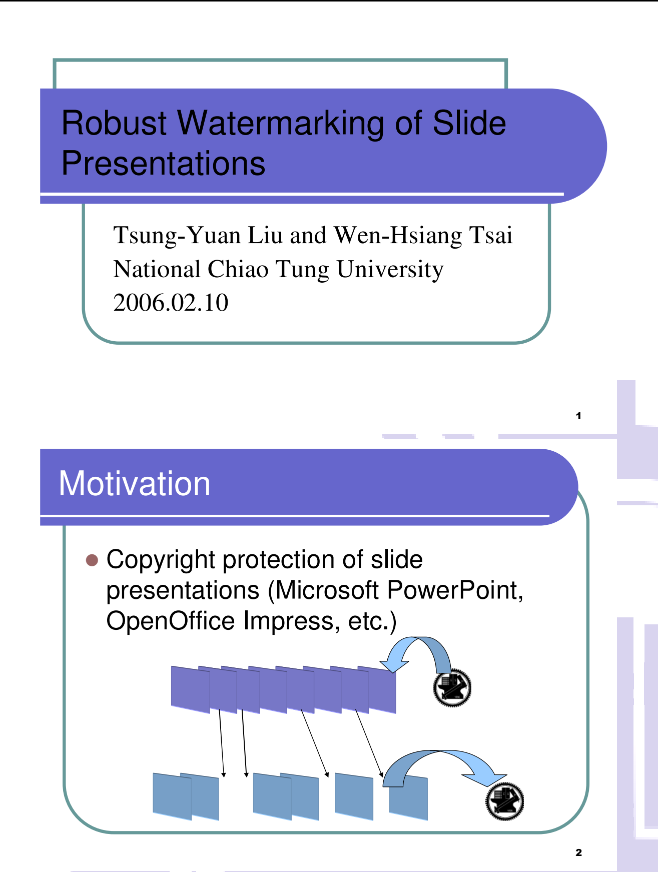
The proposed method is also resilient against removals of slides or slide contents, as long as sufficient watermarked contents remain. Experiments have shown that a recognizable watermark of size 64×64 can be reconstructed from approximately five watermarked slides.

Lastly, the watermark embedded using the proposed method has been proven to be robust against file format conversion attacks. Specifically, a presentation with slides watermarked using the proposed method was first saved in Microsoft PowerPoint in its PPT format. The file was then opened by another slide presentation editing software, OpenOffice Impress, and saved in the OpenDocument ODP format. The ODP format file was then reopened by OpenOffice Impress, and finally saved back into the PPT format. Figure 6.5(a) shows the first two slides of a test presentation before file format conversion, and Figure 6.5(b) shows the same two slides after the above described format conversions from PPT to ODP and back to PPT. We note that the font type (changed from *Arial* to *Times New Roman*) and the font size (changed from 42pt to 44pt) of the title on the first slide, and the drawing on the second slide, among others, were changed during the file format conversions. The watermark image embedded in the coloring of blank spaces, however, was untouched during the process and can be perfectly reconstructed.

## 6.8 Experimental Results

The proposed watermark embedding and extraction methods were implemented[15] using C$^{\#}$.NET and a series of experiments are conducted on Microsoft PowerPoint 2003. We have collected slides from the presentations of some past projects of Dr. Tsai, as well as some presentations that are available from the web [100]-[102]. The average numbers of spaces per slide in these samples range from 35 to 60. Three slides, named A, B, and C, respectively, with the characteristics listed in Table 6.1, are chosen for the experiments.

15 The implementation can be downloaded at http://sites.google.com/site/ktyliu/invisible-watermarking-in-powerpoint-by-blank-space-coloring.



## Robust Watermarking of Slide Presentations

Tsung-Yuan Liu and Wen-Hsiang Tsai
National Chiao Tung University
2006.02.10

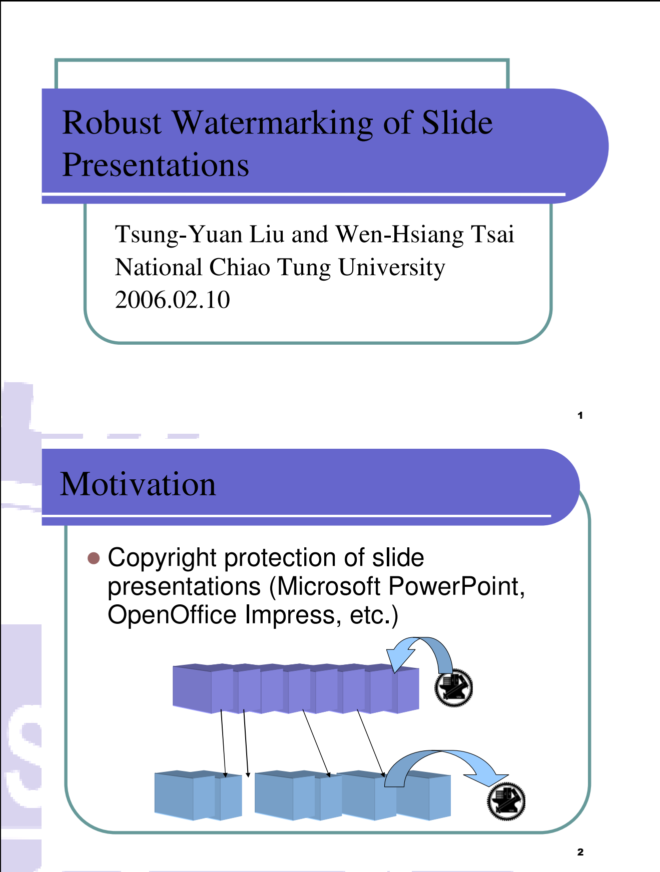
1

## Motivation

- Copyright protection of slide presentations (Microsoft PowerPoint, OpenOffice Impress, etc.)

2

(a)



## Robust Watermarking of Slide Presentations

Tsung-Yuan Liu and Wen-Hsiang Tsai
National Chiao Tung University
2006.02.10

1

## Motivation

- Copyright protection of slide presentations (Microsoft PowerPoint, OpenOffice Impress, etc.)

2

(b)

Figure 6.5. An experimental result of file format conversion. (a) Two slides in Microsoft PowerPoint. (b) The two slides after file format conversion from PPT to ODP and back.

Table 6.1. Characteristics of presentations used in the experiments.

| | A | B | C |
|---|---|---|---|
| Number of slides | 35 | 28 | 51 |
| Total number of spaces | 2,086 | 1,029 | 2,605 |
| Average number of spaces per slide | 59.6 | 36.8 | 51.1 |
| Maximum number of spaces in a slide | 352 | 159 | 339 |
| Minimum number of spaces in a slide | 0 | 1 | 1 |
| Standard deviation of spaces per slide | 73.0 | 37.4 | 53.7 |

In the experiments, three logo images, each of size 64×64, are used as watermarks. Each of them was divided into 256 blocks of 4×4 pixels each, and individually embedded into the slides of the three presentations with different security keys using Algorithm 6.1. A presentation was then constructed by drawing slides randomly from the watermarked presentations. Specifically, *N* slides were drew randomly from each of the three presentations and then combined to form an experimental presentation that contains 3*N* slides. The three watermark logos were then extracted from the experimental presentation with the three respective keys in turn using Algorithm 6.4.

The number of pixels that were correctly reconstructed in each of the three extracted watermark logos was counted, and the fractions of correct pixel extraction for the three images were recorded for each trial. This process was repeated 10,000 times for each value of *N* ranging from 1 to 19, and the average correct coverages of the three extracted watermarks are plotted in Figure 6.6. To reconstruct a recognizable extracted watermark with at least 50% correct coverage, 3, 5, and 4 slides are required from presentations A, B, and C, respectively; and to reconstruct one with 80% correct coverage, 7, 11, and 8 slides are required from A, B, and C, respectively.

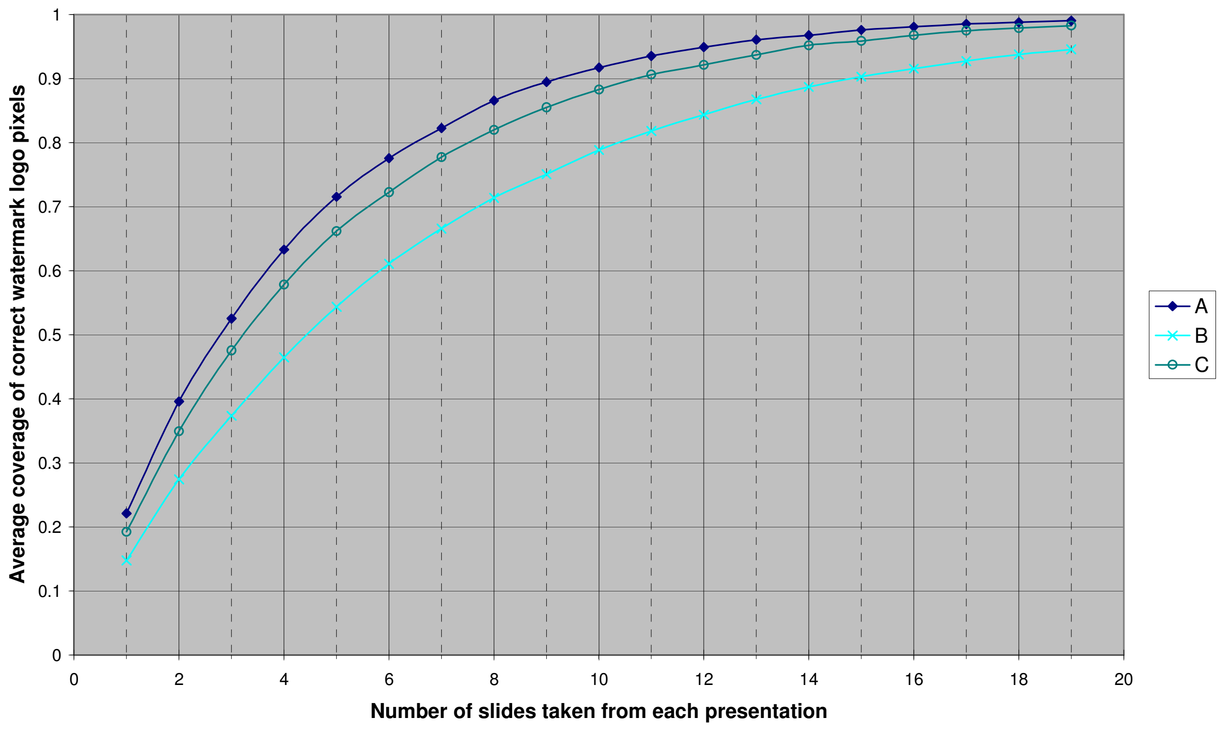


Figure 6.6. Plot of average correct watermark pixel extractions from presentations constructed from randomly drawn slides.

Figure 6.7 shows one result of the three extracted watermarks for *N* ranging from 3 to 10. The result is imperfect because of the large variations in the number of spaces in each slide. Specifically, some slides in the presentations used in the experiments contain more than one hundred spaces. The selection of just a few of these slides will result in perfect reconstruction of the watermark image. For example, the presence of a slide from presentation C that contains 339 spaces (shown in Figure 6.1) alone would allow for perfect reconstruction of the embedded watermark.

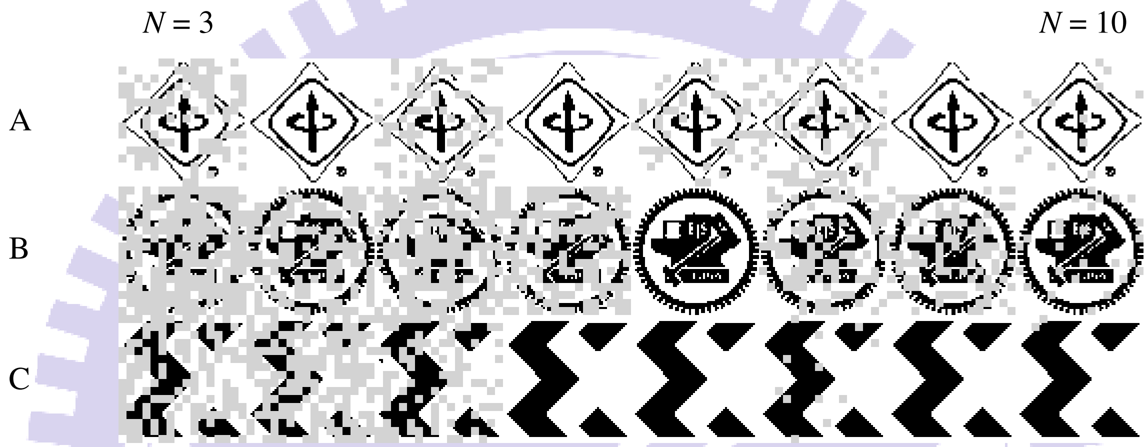


Figure 6.7. An experimental result of the three extracted watermarks with *N* ranging from 3 to 10.

On the other hand, if slides that contain less than ten spaces were picked during a trial of the experiment, then many blocks of the reconstructed watermark image will be missing. We note that the randomly constructed presentation contains slides watermarked with a key different to the one used for extraction, and can thus include incorrect data for these missing blocks. However, since such erroneous blocks do not follow the expected sequence specific for the extraction key, they are effectively filtered out by the proposed weighted voting technique, as observed in the figure. The average percentages of the watermark that was incorrectly recovered fall in the range of 0.02%~0.24% in the experiments.

The plot in Figure 6.6 was normalized by multiplying the number of slides taken from each of the presentation by the average number of spaces per slide. The normalized plot, with the average number of spaces taken from each presentation as the x-axis, is shown in Figure 6.8. It is clear that the performance of the proposed method is relatively insensitive to the properties of the slides in a presentation. The estimated coverage of the extracted watermark derived as Equation (6.1) is also plotted in the figure for comparison.

To achieve a correct coverage of 50% and 80% respectively, approximately 165 and 385 spaces are required respectively according to the figure, which are close to the theoretical estimates of 178 and 412. The actual experimental results are slightly better than the theoretical estimates because we assumed that all spaces were drawn randomly for the estimate, whereas in the experiments a whole slide that contains spaces in sequence was taken at a time. As observed in the experimental results, the interferences between the different sets of watermarked slides have been eliminated by the application of the proposed weighted voting technique.

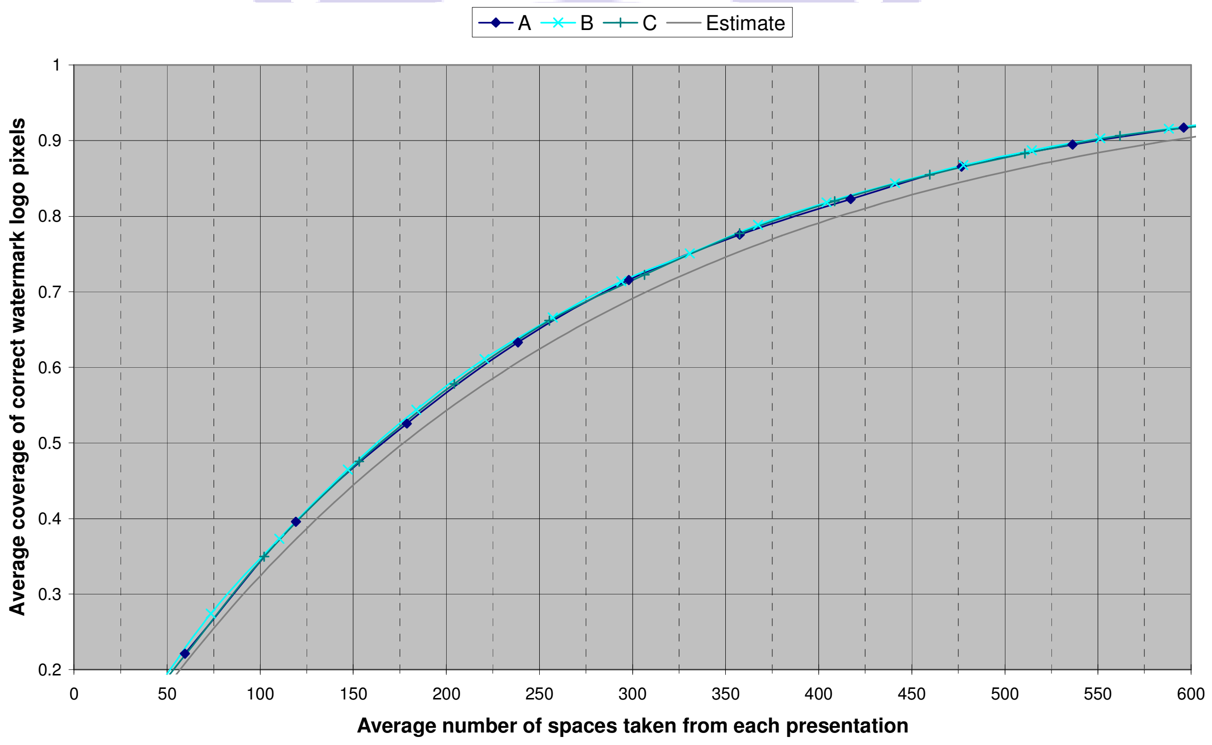


Figure 6.8. Normalized plot of average correct watermark pixel extractions from presentations constructed from randomly drawn slides.

## 6.9 Summary

In this chapter we described a novel method for embedding watermark images imperceptibly into the slides of presentations. The watermarked presentation is visually indistinguishable from the original version, and is resilient against many common editing operations. Specifically, the watermarked presentation is resilient against insertions, removals, and reordering of slides, copying and pasting of slides, changes to the slide design templates, and file format conversions.

Furthermore, if slides taken from multiple presentations that have been watermarked using different keys are combined into a single presentation, each of the previously embedded watermark images can be individually extracted correctly with the respective keys using the proposed method.

The watermark embedding and extraction methods have been tested using the popular presentation software Microsoft PowerPoint, and good experimental results demonstrate the feasibility and resilience of the proposed method. On average, five slides taken from a watermarked presentation using a certain security key is sufficient to extract a recognizable watermark of size 64×64.

# Chapter 7
# Data Hiding in Graphic Drawings by Structures of Object Groupings

## 7.1 Overview of Proposed Method

Drawings such as flowcharts, network topologies, and floor plans are commonly seen in office documents. Figure 2.4 (repeated below as Figure 7.1 for convenience) shows an example of a floor plan drawing, where the shapes representing desks, chairs, servers, walls, doors, etc. all come from standard stencils. A drawing of this kind can contain numerous objects, and for ease of manipulation, drawing editing applications allow objects to be grouped together such that each group can be manipulated as a unit. Each group can then be translated, scaled, rotated, mirrored, or colored as a whole. It is also possible for the groupings to be nested, that is, a group can contain several groups of objects.

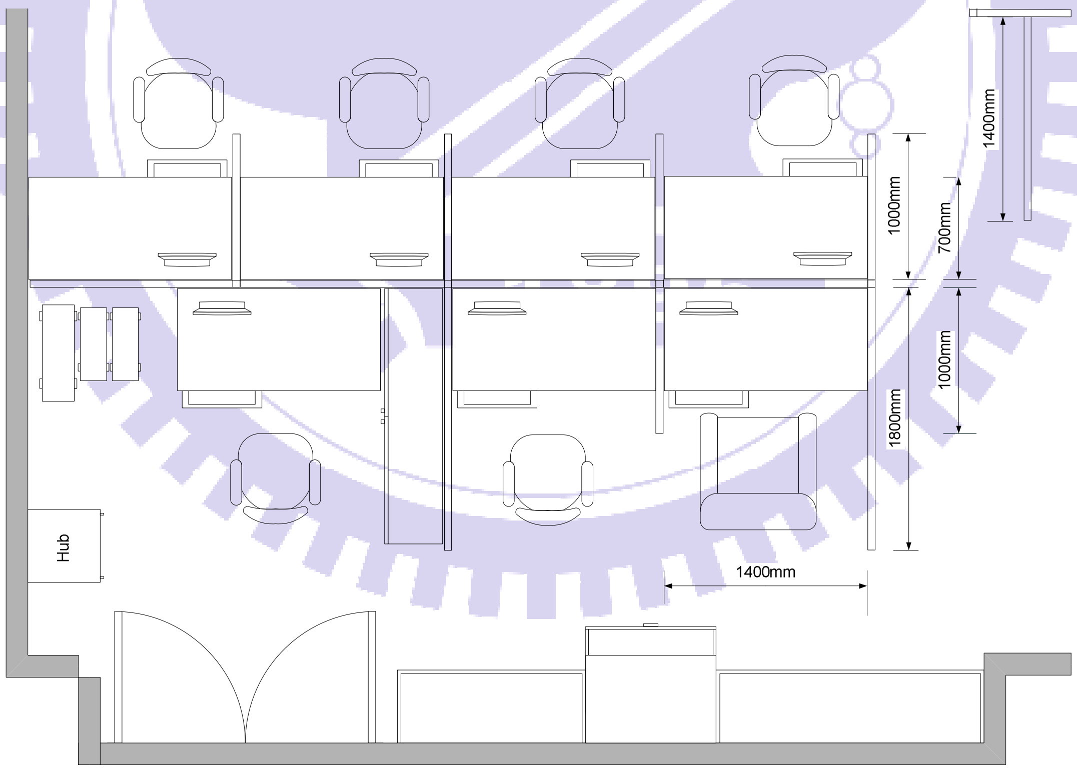


Figure 7.1. A floor plan diagram of an office composed of different objects from stencils.

In this chapter a new approach to embedding data into drawings is proposed by manipulating the structure of object groupings. Such a structure can be represented by a tree, as illustrated in Figure 7.2, where the internal nodes are the groups, and the leaves are the objects. In the figure, for example, Group 4 contains two simple objects while Group 1 includes two simple objects as well as two smaller groups. It is the use of such different combinations of the object groupings that fulfills information embedding in the proposed approach.

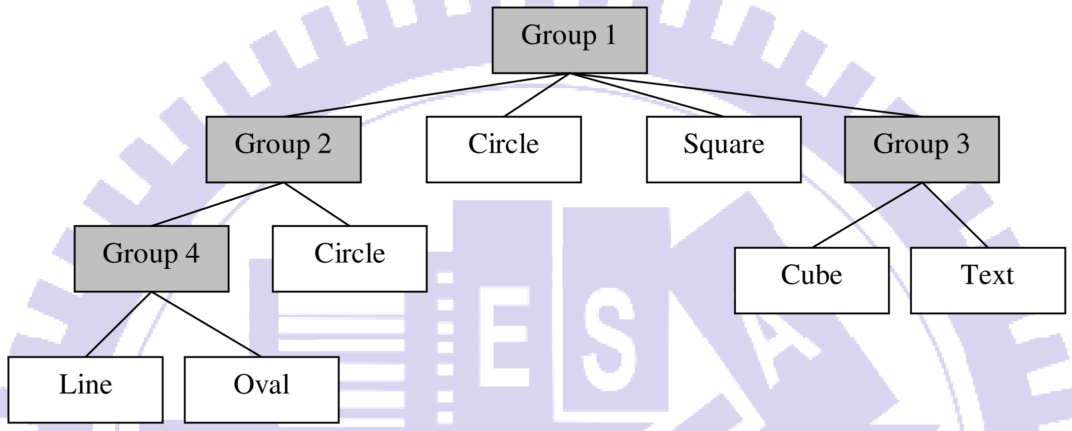


Figure 7.2. Illustration of object groupings for data embedding in a drawing.

Compared to previous data hiding techniques for vector drawings that are described in Section 2.4, the proposed method has several merits.

1. The method can be used to embed multiple data bits with a blind extraction capability, allowing for different types of data hiding applications, whereas some techniques require the use of the *original* media (called *non-blind* methods) or can only embed watermarks with no message data (called *zero-bit watermarking* methods) [60], [63].
2. Manipulations of object groupings do not change the visual appearance of a drawing at all, whereas most other techniques degrade the quality of a drawing [60]-[65].
3. Any collection of shapes, lines, or text blocks that can be grouped together may be used by the proposed method for data embedding, whereas many other techniques can only be applied to specific drawing objects such as polylines, polygons, B-spline curves, etc. [60]-[62], [65]-[66].
4. The method can be used to embed information in any graphical file format that supports nested grouping of objects, for example, AutoCAD drawings, Visio drawings, and PowerPoint presentation files.

## 7.2 Proposed Data Embedding Process

The basic idea of the proposed data embedding process is to determine the inter-object distances for all objects in the drawing, sort the distances between pairs of objects, and then group the pairs of objects in turn or left them ungrouped, depending on the bits to be embedded. Since the grouping of objects can be nested, for simplicity we will refer to a simple object or a group of objects both as an object. That is, a drawing is composed of a collection of objects, where some objects are composed of smaller constituent objects. It is assumed in this study that the cover drawing $D$ consists of simple objects and no groups. The details of the proposed data embedding process are described in the algorithm below.

**Algorithm 7.1: data embedding by structure of object groupings.**

***Input*:** a drawing with objects $D = \{o_1, o_2, \ldots, o_L\}$ and a bit string $S = b_1b_2\ldots b_N$ to be embedded.

***Output*:** a stego-drawing $D'$ with an appropriate structure of object groupings representing $S$.

***Steps:***

1. Set $D'$ to be equal to $D$ initially.
2. Create an auxiliary *helper set* $P$ and set it to be empty initially.
3. Take in order a bit $b_i$ from the input bit string $S$, where $1 \le i \le N$, and perform the following steps.
   a. Calculate the distances between every pair of objects in $D'$ that are *not* in $P$.
   b. Find the two objects $o_j$ and $o_k$ in $D'$ such that their distance is the smallest among those of all object pairs in $D'$ that are *not* in $P$.
   c. If $b_i = 1$, then group $o_j$ and $o_k$ in $D'$ together; otherwise, add the pair $(o_j, o_k)$ to $P$ as an ungrouped one.
4. Take the final $D'$ with the resulting structure of object groupings as the output stego-drawing.

The purpose of creating the set $P$ in the above algorithm is to record pairs of objects that are not grouped together for embedding 0's, so that these object pairs are not considered further. As an example, supposed that we want to embed the bits 1010010011 into the drawing shown in Figure 7.3 with the inter-object distances listed in Table 7.1.

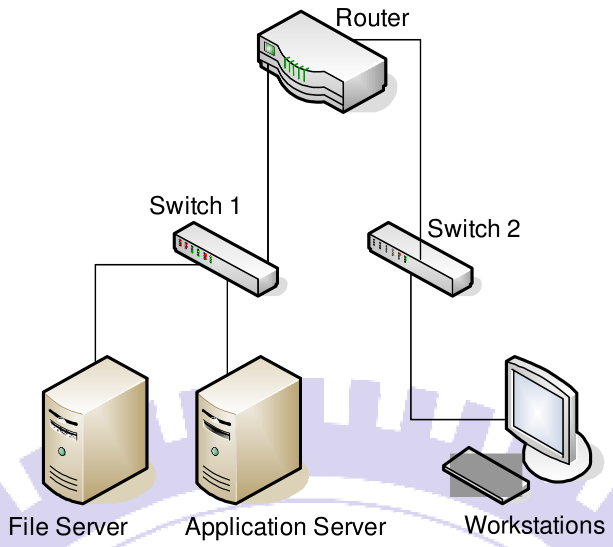


Figure 7.3. A simple drawing used as an example for embedding by object grouping.

Table 7.1. Distances between all pairs of objects in Figure 7.3.

| | Workstations | Router | Switch 2 | Switch 1 | File Server |
|---|---|---|---|---|---|
| Application Server | 0.9139 | 1.7557 | 0.9903 | 0.4134 | 0.2791 |
| File Server | 1.9525 | 2.1690 | 1.7464 | 0.8552 | |
| Switch 1 | 1.5705 | 1.0851 | 0.5998 | | |
| Switch 2 | 0.4977 | 0.7566 | | | |
| Router | 1.9052 | | | | |

We find that the objects “File server” and “Application server” are the closest two objects (with distance 0.2791), and is so the first pair considered. Since the first bit to be embedded is “1,” the two objects are grouped together to form a new object, called Group 1, according to Step 3c of the algorithm. The closest object pair in the resulting drawing is then Group 1 and the object “Switch 1” (with distance 0.4134). These two objects are *not* grouped in order to embed a “0,” and the pair is recorded in the set *P* so that they are not considered further for grouping. The objects “Switch 2” and “Workstations” (with distance 0.4977) are then considered, and grouped to form Group 2 to embed a “1,” and so on. The resulting structure of object groupings after embedding 1010010011 is shown in Figure 7.4.

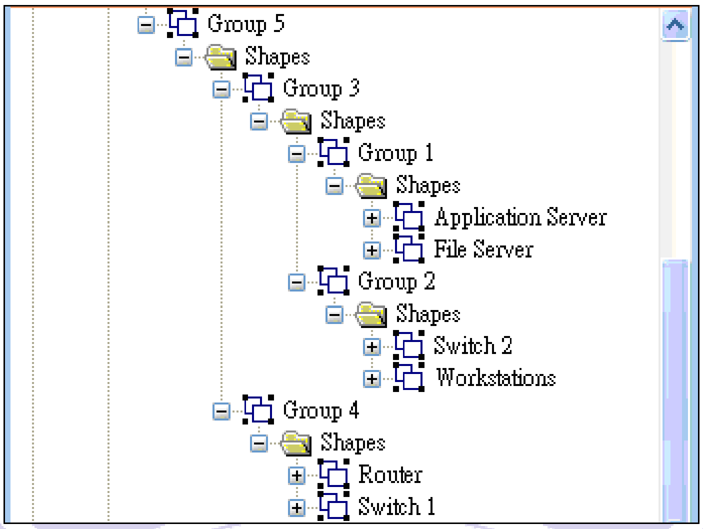


Figure 7.4. Resulting structure of object groupings of Figure 7.3 after embedding 1010010011.

## 7.3 Analysis of Data Embedding Capacity

In Algorithm 7.1, when two objects are grouped into one for every bit of 1 embedded (Step 3c), the number of objects in the stego-drawing $D'$ decreases by one. The maximum number of 1's that can be embedded using the proposed method is thus $L - 1$ where $L$ is the number of objects in the input drawing $D$.

On the other hand, the method can embed 0's more efficiently. Specifically, it can be seen from the following analysis of performing the steps of Algorithm 7.1 that for a drawing with $L$ objects, the maximum number of bits that can be embedded is $(L - 1)^2$ when the input data string is of the form:

$$\underbrace{00...0}_{\frac{L(L-1)}{2}-1}1\underbrace{00...0}_{(L-2)-1}1\underbrace{00...0}_{(L-3)-1}1...1\underbrace{00}_{3-1}1\underbrace{0}_{2-1}11. \tag{7.1}$$

***Performance analysis of* Algorithm 7*.1 for embedding the maximum number of bits.***

1. All object pairs except "the pair $G_1$ with the farthest *in-pair distance*" are ungrouped and added to the set $P$ to embed the first $\left(\frac{L(L-1)}{2}-1\right)$ 0's, where the term "in-pair distance" means the distance between the two objects in an object pair.
2. $G_1$ is taken as Group 1 to embed the first "1" in the data string, which results in $L - 2$ new distance relationships between Group 1 and the remaining $L - 2$ objects.
3. All object pairs, each with Group 1 and an object of the remaining $L - 2$ ones, except the pair $G_2$ with the farthest in-pair distance are ungrouped and added to the set $P$ to embed the next $[(L - 2) - 1]$ 0's.

4. $G_2$ is taken as Group 2 to embed the second "1" in the data string, resulting in $L - 3$ new distance relationships.
5. Steps 3 and 4 above are repeated in a similar way for the remaining objects and groups until no more objects can be considered for grouping.

Accordingly, the maximum number of bits that can be embedded is

$$\{[L(L-1)/2-1]+1\}+\{[(L-2)-1]+1\}+\{[(L-3)-1]+1\}+\ldots+\{[2-1]+1\}+1$$
$$= L(L-1)/2+(L-2)+(L-3)+\ldots+1$$
$$= L(L-1)/2+(L-1)(L-2)/2$$
$$= (L-1)^2,$$

as mentioned previously.

On the other hand, the expected number of random bits (equal occurrence probabilities of 0's and 1's) that can be embedded by the proposed method using a drawing with $L$ objects is smaller than $2(L-1)$, as discussed now.

First, as mentioned previously, at most $(L-1)$ 1's can be embedded for a drawing with $L$ objects. Also, from the above performance analysis of the proposed algorithm for embedding the maximum number of bits, we see that at least one "0" can be embedded for each "1" embedded except the last one. If the input string includes 0's and 1's alternatively as in the extremely random case, then exactly $2(L-1)-1$ bits can be embedded. In real cases, 1's may appear *consecutively*, such that objects in a drawing will be exhausted *faster* than appropriate numbers of 0's are embedded, resulting in less 0's being embedded when compared with the extremely random case. In short, for the average case, the number of random bits that can be embedded with a drawing containing $L$ objects is roughly $2(L-1)-1 \approx 2L$ if $L$ is large enough.

## 7.4 Proposed Data Extraction Process

The process for extracting the data embedded in a stego-drawing using Algorithm 7.1 is described below. Basically, the algorithm first removes the structure of object groupings to recover the original cover drawing. It then uses the same procedure as that of data embedding to gradually reconstruct the same structure of object groupings, and checks in the meantime the object grouping structure in the stego-drawing to determine the previously embedded bits one by one.

**Algorithm 7.2: data extraction from structure of object groupings.**

***Input***: a stego-drawing $D'$ with a certain structure of object groupings generated by Algorithm 7.1.

***Output***: a bit string $S = b_1b_2\ldots b_N$ extracted from the structure of object groupings in $D'$.

***Steps:***

```
1. Ungroup all existing object groups in $D'$ to recover the original drawing $D = \{o_1, o_2, \ldots, o_L\}$.
2. Create an auxiliary helper set $P$ and set it to be empty initially.
3. Initialize an empty bit string $S$.
4. Extract a bit $b_i$ and append it to the end of $S$, where $1 \le i \le N$, by performing the following steps.
   a. Calculate the distances between every pair of objects in $D$ that are not in $P$.
   b. Find the two objects $o_j$ and $o_k$ in $D$ such that their distance is the smallest among those of all object pairs in $D$ that are not in $P$.
   c. Check if the object pair $(o_j, o_k)$ is a group in the grouping structure of the original stego-drawing $D'$: if so, then set $b_i$ to be 1 and take $(o_j, o_k)$ as a group in $D$; otherwise, set $b_i$ to be 0 and add the pair $(o_j, o_k)$ to $P$ as an ungrouped one.
5. Take the final $S$ as the desired output bit string.
```

In the above algorithm, it is assumed that the number of bits $N$ in the embedded bit string is either previously known or determinable during data extraction in a certain way, like prefixing the data bit string $S$ with a fixed-length bit segment that contains the value $N$, similar to that described in Section 3.3.

## 7.5 Possible Data Hiding Applications

The proposed method uses relative inter-object distances as the basis for forming the structure of object groupings. It can be figured out that the technique is robust to attacks such as translation, scaling, rotation, and mirroring of the drawing as a whole since these transformations do not alter the inter-object distances. Based on this property, we describe below two possible data hiding applications using the proposed method.

First, the proposed method can be used for authenticating a drawing by embedding a random-key controlled bit string with $(L - 1)$ 1's as an authentication signal such that all objects in the drawing are grouped together, forming a structure of mutually-validating object groups. If an attacker attempts to modify the stego-drawing by changing some of the object groupings, moving some of the objects around, or adding or removing one or more objects, then some of the inter-object distances in the drawing will be changed. The data extracted from such a drawing will thus be different from the expected embedded authentication signal and tampering with the drawing be detected.

It is noted that the above method by itself cannot detect attacks where an object is replaced by another with the same dimension, or where an object's internal properties (such as color and caption) are modified. For such cases, the proposed method can be combined with other authentication mechanisms which cover the objects' properties. For example, one simple technique is to create an encrypted hash value [91] from all the objects' properties and then embed this hash value as an authentication signal using the proposed algorithms, resulting in the combined method of object-grouping and object-property encryption techniques. Tampering with an object's property in a stego-drawing can then be detected since the authentication signal extracted from such a tampered drawing using the proposed algorithms will be different from that re-calculated directly from the changed objects' properties

Second, since communications via drawings such as flowcharts are common and the proposed data hiding procedure does not affect the visual appearance of the drawing, the method is suitable for the covert communication purpose as well. Specifically, a secret message in the form of a bit string can be embedded into a cover drawing imperceptibly using the proposed algorithms. Also, grouping of objects for object manipulations in a drawing is a subjective choice by the author and usually does not follow a predictable rule, making automatic steganalysis difficult.

To face the common assumption made in covert communication that the data hiding algorithms used are known to the public, it is suggested to randomize the secret message in advance by some symmetric encryption algorithm before it is taken as input to the proposed data embedding process.

## 7.6 Experimental Results

A series of experiments were conducted on some drawings created with Microsoft Visio 2003, a popular package for drawing flowcharts, network diagrams, floor plans, etc. Two examples of them are shown in Figure 7.1 and Figure 7.5. The proposed method can be conveniently implemented[16] in Visio, which supports nested grouping of objects as well as functionality that allows distances between any two objects to be computed [103]. Three different types of drawings as listed in Table 7.2 were tested to demonstrate the generic applicability of the proposed method. The table lists the number of objects available for grouping in each drawing, and the average number of random bits embeddable over ten independent trials. The average number of bits embeddable is approximately twice the number of objects in the drawing, which matches the theoretical prediction mentioned in Section 7.3.

The stego-drawings were then attacked by translation, scaling, and rotation operations in the experiments. Specifically, we translated, scaled, and/or rotated all the objects in the drawing simultaneously in the attacks and applied the algorithms subsequently. The results show that the bits embedded in the object grouping structures survive these attacks.

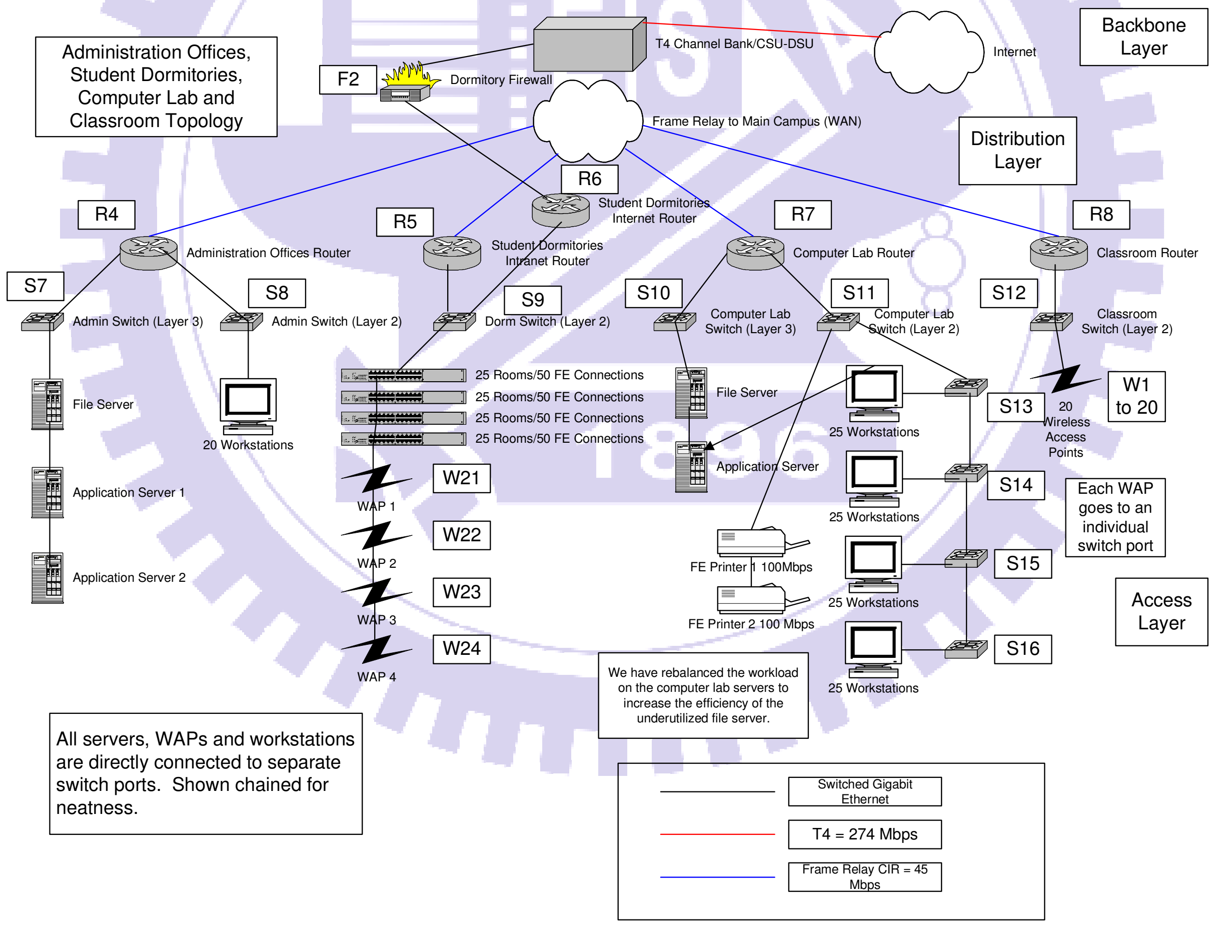

Administration Offices, Student Dormitories, Computer Lab and Classroom Topology
F2
Dormitory Firewall
T4 Channel Bank/CSU-DSU
Internet
Backbone Layer
Frame Relay to Main Campus (WAN)
Distribution Layer
R6
Student Dormitories Internet Router
R4
Administration Offices Router
R5
Student Dormitories Intranet Router
R7
Computer Lab Router
R8
Classroom Router
S7
Admin Switch (Layer 3)
S8
Admin Switch (Layer 2)
S9
Dorm Switch (Layer 2)
S10
Computer Lab Switch (Layer 3)
S11
Computer Lab Switch (Layer 2)
S12
Classroom Switch (Layer 2)
File Server
20 Workstations
Application Server 1
Application Server 2
25 Rooms/50 FE Connections
25 Rooms/50 FE Connections
25 Rooms/50 FE Connections
25 Rooms/50 FE Connections
W21
WAP 1
W22
WAP 2
W23
WAP 3
W24
WAP 4
File Server
Application Server
25 Workstations
25 Workstations
25 Workstations
25 Workstations
FE Printer 1 100Mbps
FE Printer 2 100 Mbps
S13
S14
S15
S16
20 Wireless Access Points
W1 to 20
Each WAP goes to an individual switch port
Access Layer
We have rebalanced the workload on the computer lab servers to increase the efficiency of the underutilized file server.
All servers, WAPs and workstations are directly connected to separate switch ports. Shown chained for neatness.
Switched Gigabit Ethernet
T4 = 274 Mbps
Frame Relay CIR = 45 Mbps

Figure 7.5. A network layout diagram used in the experiments (source: UCF).

[16] A prototype implementation can be downloaded at http://sites.google.com/site/ktyliu/data-hiding-by-structures-of-object-grouping.

Table 7.2. Experimental results of embedding capacity for different drawings.

| | Type of drawing | Number of objects | Bits embeddable |
|---|---|---|---|
| A | Network topology | 113 | 235.0 |
| B | Office layout | 78 | 156.4 |
| C | Flowchart | 44 | 82.0 |

## 7.7 Summary

In this chapter, a new data hiding method has been proposed, which embeds message data imperceptibly into the structure of object groupings in a drawing, in contrast with prior works that alter objects themselves for data hiding applications. The proposed method is generic and can be applied to a variety of drawings, including flowcharts, network diagrams, circuit schematics, floor plans, etc. The method creates a structure of object groupings based on the data to be embedded as well as inter-object distances in the drawing, yielding a stego-drawing that is robust against attacks such as translation, scaling, rotation, and mirroring operations. The proposed method can be applied to different data hiding applications, such as drawing authentication and covert communication. Finally, experiments conducted with Microsoft Visio documents confirm the feasibility of the proposed method.

# Chapter 8

# Generic Lossless Visible Watermarking – A New Approach

## 8.1 Introduction

In this chapter, a new method for lossless visible watermarking is proposed by using appropriate *compound mappings* that allow mapped values to be controllable. The mappings are proved to be *reversible* for lossless recovery of the original image. The approach is *generic*, leading to the possibility of embedding different types of visible watermarks into cover images. Two applications of the proposed method are demonstrated, where opaque monochrome watermarks and non-uniformly translucent full-color ones are respectively embedded into color images. More specific compound mappings are also created and proved to be able to yield visually more distinctive visible watermarks in the watermarked image. To the best knowledge of the authors, this is the first method ever proposed for embedding *removable translucent full-color watermarks* which provide better advertising effects than traditional monochrome ones. It is also demonstrated in this study that the embedding of a watermarked image in a Microsoft Word or a Microsoft PowerPoint document does not affect the lossless recoverability of the original image.

In the remainder of this chapter, the proposed method for deriving one-to-one compound mappings is described in Section 8.2. Related lemmas and theorems are also proved and security protection measures described. Applications of the proposed method for embedding opaque monochrome and translucent color watermarks into color images are described in Sections 8.3 and 8.4, respectively. In Section 8.5, the specific compound mapping for yielding more distinctive visible watermarks is described. In Section 8.6, experimental results are presented to demonstrate the effectiveness of the proposed method. Finally, a summary is included in Section 8.7.

## 8.2 Proposed New Approach to Lossless Visible Watermarking

In this section, we describe the proposed approach to lossless reversible visible watermarking, based on which appropriate one-to-one compound mappings can be designed for embedding different types of visible watermarks into images. The original image can be recovered losslessly from a resulting watermarked image by using the corresponding reverse mappings.

### 8.2.1 Reversible One-To-One Compound Mapping

First, we propose a generic one-*to-one compound mapping* $f$ for converting a set of numerical values $P = \{p_1, p_2, \ldots, p_M\}$ to another set $Q = \{q_1, q_2, \ldots, q_M\}$, such that the respective mapping from $p_i$ to $q_i$ for all $i = 1, 2, \ldots, M$ is *reversible*. Here, for the copyright protection applications investigated in this study, all the values $p_i$ and $q_i$ are image pixel values (grayscale or color values). The compound mapping $f$ is governed by a one-to-one function $F_x$ with one parameter $x = a$ or $b$ in the following way:

$$q = f(p) = F_b^{-1}(F_a(p)) \tag{8.1}$$

where $F_x^{-1}$ is the inverse of $F_x$ which, by the one-to-one property, leads to the fact that if $F_a(p) = p'$, then $F_a^{-1}(p') = p$ for all values of $a$ and $p$. On the other hand, $F_a(p)$ and $F_b(p)$ generally are set to be *unequal* if $a \neq b$.

The compound mapping described by (8.1) is indeed *reversible*, that is, $p$ can be derived exactly from $q$ using the following formula:

$$p = f^{-1}(q) = F_a^{-1}(F_b(q)), \tag{8.2}$$

as proved below.

**Lemma 8.1 (reversibility of compound mapping).** If $q = F_b^{-1}(F_a(p))$ for any one-to-one function $F_x$ with a parameter $x$, then $p = F_a^{-1}(F_b(q))$ for any values of $a$, $b$, $p$, and $q$.

***Proof.*** Substituting (8.1) into $F_a^{-1}(F_b(q))$, we get

$$F_a^{-1}(F_b(q)) = F_a^{-1}(F_b(F_b^{-1}(F_a(p)))).$$

By regarding $F_a(p)$ as a value $c$, the right-hand side becomes $F_a^{-1}(F_b(F_b^{-1}(c)))$, which, after $F_b$ and $F_b^{-1}$ are cancelled out, becomes $F_a^{-1}(c)$. But $F_a^{-1}(c) = F_a^{-1}(F_a(p))$, which is just $p$ after $F_a$ and $F_a^{-1}$ are cancelled out. That is, we have proved $p = F_a^{-1}(F_b(q))$. □

As an example, if $F_x(p) = xp + d$, then $F_x^{-1}(p') = (p' - d)/x$. Thus

$$q = F_b^{-1}(F_a(p)) = F_b^{-1}(ap + d) = (ap + d - d)/b = ap/b.$$

And so, we have

$$F_a^{-1}(F_b(q)) = F_a^{-1}(b(ap/b) + d) = F_a^{-1}(ap + d) = [((ap + d) - d)/a] = (ap/a) = p,$$

as expected by Lemma 8.1.

### 8.2.2 Lossless Visible Watermarking Scheme

Based on Lemma 8.1, we will now derive the proposed generic lossless visible watermarking scheme in the form of a class of one-to-one compound mappings, which can be used to embed a variety of *visible watermarks* into images. The embedding is reversible, that is, the watermark can be removed to recover the original image losslessly. For this aim, a preliminary lemma is first described as follows.

**Lemma 8.2 (preference of compound-mapped value $q$).** It is possible to use the compound mapping $q = F_b^{-1}(F_a(p))$ to convert a numerical value $p$ to another value close to a *preferred* value $l$.

***Proof.*** Let $F_x(p) = p - x$ where $x$ is the parameter for $F$. Then $F_x^{-1}(p') = p' + x$. Also, let $a = p - \varepsilon$ and $b = l$ where $\varepsilon$ is a small value. Then, the compound mapping $F_b^{-1}(F_a(p))$ of $p$ yields $q$ as

$$q = F_b^{-1}(F_a(p)) = F_b^{-1}(p - a) = F_b^{-1}(\varepsilon) = \varepsilon + b = \varepsilon + l,$$

which means that the value $q$ is close to the preferred value $l$. □

The above lemma relies on two assumptions. The first is that $a$ is close to $p$, or equivalently, that $a = p - \varepsilon$. The reason why we derive the above lemma for $a = p - \varepsilon$ instead of for $a = p$, is that in the reverse mapping we want to recover $p$ from $q$ *without knowing* $p$, which is a requirement in the applications of reversible visible watermarking investigated in this study. Although the value of $p$ cannot be known in advance for such applications, it can usually be estimated, and we will describe some techniques for such estimations in the subsequent sections.

The second assumption is that $F_x(p)$ yields a small value if $x$ and $p$ are close. Though the basic difference function $F_x(p) = p - x$ used in the above proof satisfies this requirement for most cases, there is a possible problem where the mapped value may exceed the range of valid pixel values for some values of $a$, $b$, and $p$. For example, when $a = 255$, $b = 255$, and $p = 253$, we have $q = 255 - 253 + 255 = 257 > 255$. It is possible to use the standard modulo technique (i.e., taking $q = 257_{\text{mod } 256} = 1$) to solve this issue; however, such a technique will make $q$ far from the desired target value of $b$, which is 255. Nevertheless, we will show in Section 3 that using such a standard modulo function, $F_x(p) = (p - x)_{\text{mod } 256}$, can still yield reasonable experimental results. Furthermore, we show in Section 8.5 a more sophisticated one-to-one function that is free from such a wraparound problem.

By satisfying the above two requirements, the compound mapping yields a value $q$ that is close to the desired value $l$. We now prove a theorem about the desired lossless reversible visible watermarking in the following.

**Theorem 8.1 (lossless reversible visible watermarking).** There exist one-to-one compound mappings for use to embed into a given image $I$ a visible watermark $Q$ whose pixel values are close to those of a given watermark $L$, such that the original image $I$ can be recovered from $Q$ losslessly.

***Proof.*** This is a consequence of Lemmas 8.1 and 8.2 after regarding the individual pixel values in $I$, $L$, and $Q$ respectively as those of $p$, $l$, and $q$ mentioned in Lemma 8.2. And it is clear by Lemma 8.1 that the value $p$ can be recovered losslessly from the mapped value $q$ which is derived in Lemma 8.2. □

The above discussions are valid for embedding a watermark in a grayscale image. If color images are used both as the cover image and the watermark, we can apply the mappings to each of the color channels to get multiple independent results. The resulting visible watermark is the composite result of the color channels.

Based on Theorem 8.1, the proposed generic lossless reversible visible watermarking scheme with a given image $I$ and a watermark $L$ as input is described as an algorithm as follows.

**Algorithm 8.1: generic visible watermark embedding.**
***Input*:** an image $I$ and a watermark $L$.
***Output*:** watermarked image $W$.
***Steps*:**

1. Select a set $P$ of pixels from $I$ where $L$ is to be embedded, and call $P$ a *watermarking area*.
2. Denote the set of pixels corresponding to $P$ in $W$ by $Q$.
3. For each pixel $X$ with value $p$ in $P$, denote the corresponding pixel in $Q$ as $Z$ and the value of the corresponding pixel $Y$ in $L$ as $l$, and conduct the following steps.
    a. Apply an estimation technique to derive $a$ to be a value close to $p$, using the values of the neighboring pixels of $X$ (excluding $X$ itself).

b. Set $b$ to be the value $l$.

c. Map $p$ to a new value $q = F_b^{-1}(F_a(p))$.

d. Set the value of $Z$ to be $q$.

4. Set the value of each remaining pixel in $W$, which is outside the region $P$, to be equal to that of the corresponding pixel in $I$.

Note that we do *not* use the information of the *original* image pixel value of $X$ itself for computing the parameters $a$ and $b$ for $X$. This ensures that identical parameter values can be calculated by the receiver of a watermarked image for the purpose of lossless image recovery.

As an example, the process performed by Step 3 of the above algorithm for a pixel is illustrated by Figure 8.1, where the north and west pixels are used to estimate the color of the center pixel. Note that the east and south pixels are not used because these pixels are covered by the watermark and unknown to the receiver. It is important to allow as many neighbors of a pixel as possible to be known by the receiver to ensure that a good estimate can be calculated for that pixel. We will describe in Section 8.4 techniques for processing pixels, which can ensure that sufficiently many neighbor colors are known by a receiver for each pixel in the watermarking area.

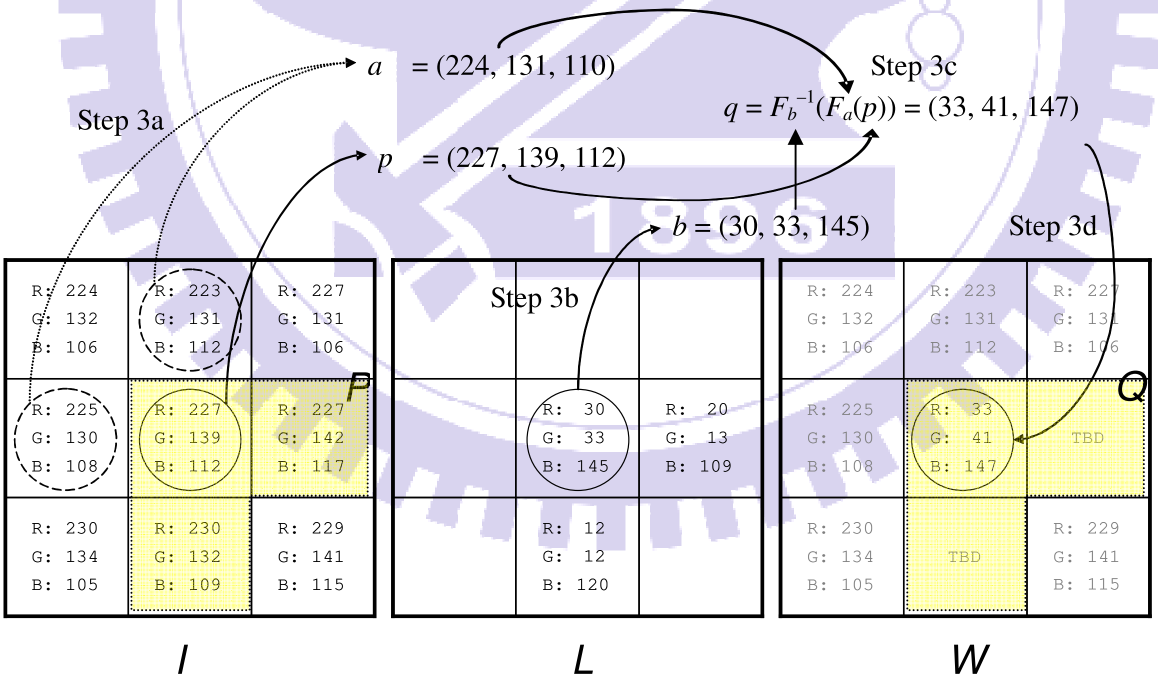


Figure 8.1. An illustration of mapping the center pixel of a 3×3 image using Algorithm 8.1. Only the mapping of the center pixel is shown for clarity; the east and south pixels are depicted as TBD (to be determined) in $W$.

The corresponding watermark removal process for a watermarked image *W* generated by Algorithm 8.1 is described as an algorithm as follows.

**Algorithm 8.2: generic watermark removal for lossless image recovery.**

***Input***: a watermarked image *W* and a watermark *L*.

***Output***: the original image *R* recovered from *W*.

***Steps:***

1. Select the same watermarking area *Q* in *W* as that selected in Algorithm 8.1.
2. Set the value of each pixel in *R*, which is outside the region *Q*, to be equal to that of the corresponding pixel in *W*.
3. For each pixel *Z* with value *q* in *Q*, denote the corresponding pixel in the recovered image *R* as *X* and the value of the corresponding pixel *Y* in *L* as *l*, and conduct the following steps.
   a. Obtain the same value *a* as that derived in Step 3a of Algorithm 8.1 by applying the same estimation technique used there.
   b. Set *b* to be the value *l*.
   c. Restore *p* from *q* by setting $p = F_a^{-1}(F_b(q))$.
   d. Set the value of *X* to be *p*.

### 8.2.3 Security Considerations

As mentioned previously, although we want legitimate users to be able to recover the original image from a watermarked one, we do not want an attacker to be able to do the same. Herein, we propose some security protection measures against illicit recoveries of original images.

First, we make the parameters *a* and *b* in the above algorithms to be dependent on certain secret keys that are known only by the creator of the watermarked image and the intended receivers. One simple technique to achieve this is to use a secret key to generate a pseudo-random sequence of numerical values and add them to either or both of *a* and *b* for the pixels in the watermarking area. This technique is hereinafter referred to as *parameter randomization*.

Another way of security protection is to make the choices of the *positions* for the pixels to be dependent on a secret key. Specifically, we propose to process *two* randomly chosen pixels (based on the security key) in *P* simultaneously as follows. Let the two pixels be denoted as $X_1$ and $X_2$ with values $p_1$ and $p_2$, respectively. The color estimates $a_1$ and $a_2$

corresponding to $X_1$ and $X_2$, respectively, are individually derived as before using their respective neighbors. The parameters $b_1$ and $b_2$ are set to be the values $l_1$ and $l_2$ of the respective watermark pixels $Y_1$ and $Y_2$. Then, instead of setting the values of the watermarked pixels $Z_1$ and $Z_2$ to be $q_1 = F_{b_1}^{-1}(F_{a_1}(p_1))$ and $q_2 = F_{b_2}^{-1}(F_{a_2}(p_2))$ as before, we *swap* the parameters and set

$$q_1 = F_{b_1}^{-1}(F_{a_2}(p_2)) \text{ and } q_2 = F_{b_2}^{-1}(F_{a_1}(p_1)).$$

This parameter exchange does not affect the effectiveness of lossless recoverability, because we can now recover the original pixel values by the following compound mappings:

$$p_1 = F_{a_1}^{-1}(F_{b_2}(q_2)) \text{ and } p_2 = F_{a_2}^{-1}(F_{b_1}(q_1)).$$

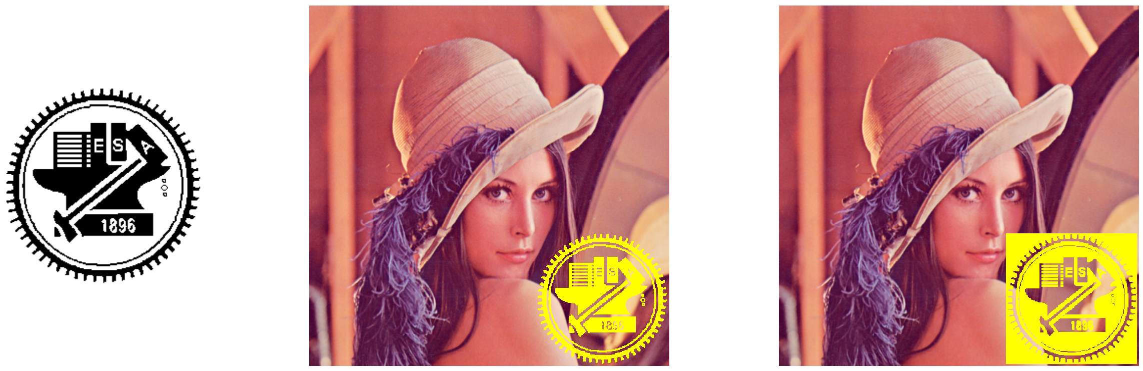
We will refer to this technique in the sequel as *mapping randomization*. We may also combine this technique with the above-mentioned parameter randomization technique to enhance the security further.

Last, the position in the image where a watermark is embedded affects the resilience of the watermarked image against illicit image recovery attempts. In more detail, if the watermark is embedded in a smooth region of the image, an attacker can simply fill the region with the background color to remove the watermark irrespective of the watermarking technique used. To counter this problem, an appropriate position should be chosen, using, for example, the adaptive positioning technique [104] when embedding a watermark. However, for ease of discussions and comparisons, we always embed a watermark in the lower right-hand corner of an image in this study.

## 8.3 Lossless Visible Watermarking of Opaque Monochrome Watermarks

As an application of the proposed generic approach to lossless visible watermarking, we describe now how we embed a losslessly-removable opaque monochrome watermark $L$ into a color image $I$ such that the watermark is *visually distinctive* in the watermarked image $W$.

First, we denote the sets of those pixels in $I$ corresponding spatially to the *black* and *white* pixels in $L$ by $P$ and $P'$, respectively. An illustration of such areas of $P$ and $P'$ is shown in Figure 8.2. We define $Q$ and $Q'$ in a similar way for the watermarked image $W$, which correspond to $P$ and $P'$, respectively.



(a) (b) (c)

Figure 8.2. An illustration of pixels in a watermark. (a) A monochrome watermark. (b) Area of $P$ (yellow pixels). (c) Area of $P'$ (yellow pixels).

Then, we adopt the simple one-to-one function $F_a(p) = p - a$, and use the same pair of parameters $a$ and $b$ for *all* mappings of pixels in $P$. Also, we apply the "modulo-256" operation to the results of all computations so that they are within the valid range of color values. Our experiments show that this method still yields reasonable results.

As to the values of parameters $a$ and $b$, we set $a$ to be the *average* of the color component values of the pixels in $P'$. This average value presumably is close to the value $p$ of pixel $X$ in $P$, fulfilling the condition $a = p - \varepsilon$ mentioned previously. To ensure that the watermark is distinctive in $W$, we do not simply embed black values for pixels in watermarking area $P$ (that is, we do not embed $l = 0$ for $P$), but set $l$ to be a value which is distinctive with respect to the pixel colors in the surrounding region $P'$. To achieve this, we set $b = l = a + 128$, which is a value *distinctive* with respect to $a$. As a result, the value of a pixel in $Q$, according to Lemma 8.2, becomes $q = F_b^{-1}(F_a(p)) = b + \varepsilon = a + 128 + \varepsilon$, meaning that the pixel values of $Q$ are also distinctive with respect to those of the surrounding pixels in $Q'$ as desired.

On the other hand, since both $a$ and $b$ are derived from $P'$ during watermark embedding, the exact same values of $a$ and $b$ can be derived during watermark removal because $Q'$ is identical to $P'$. The original image can therefore be recovered losslessly using Algorithm 8.2.

To demonstrate the effectiveness of the proposed method, in one of our experiments we embedded the watermark of Figure 8.2 (a) into the images Lena and Sailboat, respectively, and the results are shown in Figure 8.3. For security protection, we applied both the mapping randomization and the parameter randomization techniques described in Section 8.2.3. Specifically, for the latter technique we added random integer values in the range of −12 to +12 to the parameter $b$.

The images recovered by using correct keys for the parameter and mapping randomization processes are shown in Figures. 7.3(c) and 7.3(g), and those recovered with incorrect keys are shown in Figures. 7.3(d) and 7.3(h). We observe from these figures that the embedded opaque watermarks are distinctive with respect to their surroundings and can be removed completely when the input key is correct. On the contrary, when the key was incorrect, the inserted watermark cannot be removed cleanly, with noise remaining in the watermarking area.

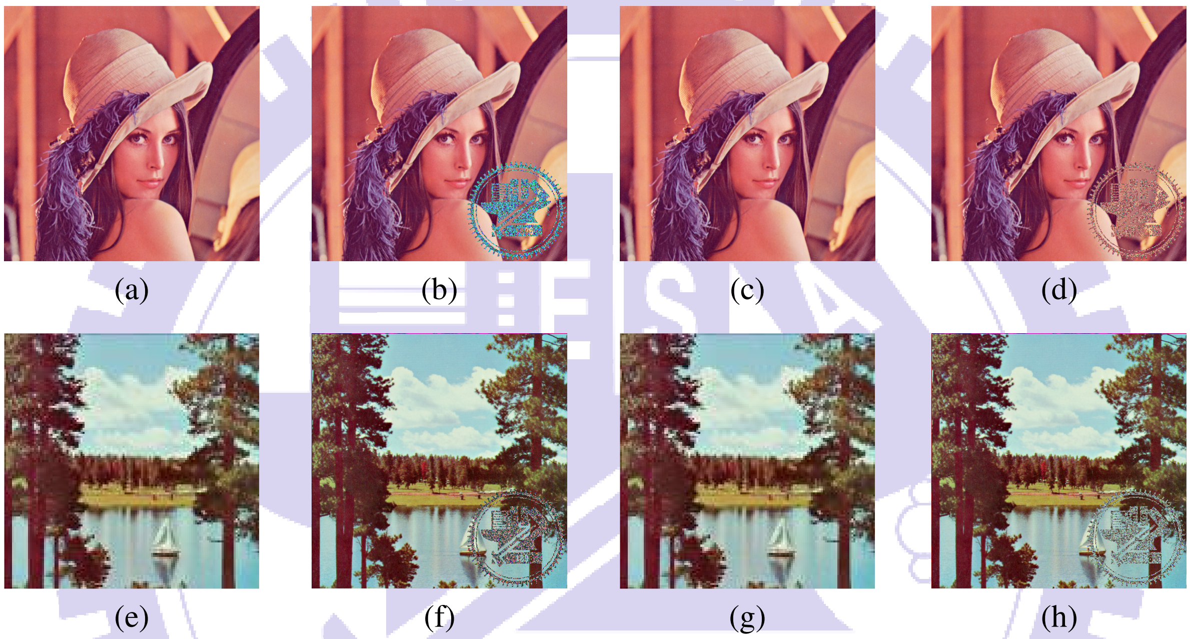


Figure 8.3. Experimental results of monochrome watermark embedding and removal. (a) Image Lena. (e) Image Sailboat. (b) and (f) Watermarked images of (a) and (e), respectively. (c) and (g) Images losslessly recovered from (b) and (f), respectively, with correct keys. (d) and (h) Images recovered from (b) and (f) with incorrect keys.

## 8.4 Lossless Visible Watermarking of Translucent Color Watermarks

As another application of the proposed approach, we describe now how we embed more complicated *translucent color watermarks*. A translucent color watermark used in this study is an arbitrary RGB image with each pixel being associated with an *alpha component value* defining its *opacity*. The extreme alpha values of 0 and 255 mean that the watermark pixel is *completely* transparent and *totally opaque*, respectively. A translucent full-color watermark is visually more attractive and distinctive in a watermarked image than a traditional transparent monochrome watermark, as mentioned previously. Such a kind of watermark can better represent trademarks, emblems, logos, etc., and thus is more suitable for the purpose of advertising or copyright declaration.

If recoverability is not an issue, we can overlay the translucent watermark over the original image with an application package like Photoshop using the standard *alpha blending* operation to obtain a watermarked image, as illustrated in Figure 2.3, repeated below as Figure 8.4 for convenience. Such an image will be called a *non-recoverable watermarked image* in the sequel, and will be used as a *benchmark* in our experiments.

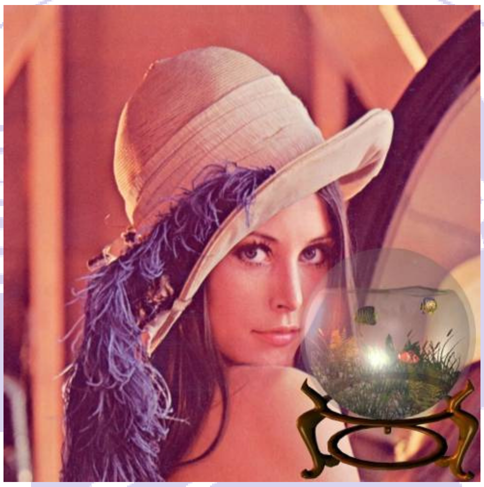


Figure 8.4. Watermarked image of Lena with a translucent image of "Globe" superimposed using alpha blending.

The proposed algorithm for embedding a translucent color watermark is similar to Algorithm 8.1 and is described below. To ensure that the parameter $a$ is close to $p$ for each pixel, we keep track of the pixels that have been *processed* throughout the embedding process. The pixels outside region $P$ need not be processed and are regarded as having been processed in the following discussion.

**Algorithm 8.3: watermark embedding of a translucent color watermark.**

***Input*****:** an image $I$ and a translucent watermark $L$.

***Output*****:** a watermarked image $W$.

***Steps:***

1. Select the watermarking area $P$ in $I$ to be the set of pixels corresponding spatially to those in $L$ which are non-transparent (with alpha values larger than zero).

2. Denote the set of pixels corresponding to $P$ in $W$ as $Q$.
3. For each pixel $X$ with value $p$ in $P$, denote the corresponding pixel in $Q$ as $Z$ and the value of the corresponding pixel $Y$ in $L$ as $l$, and conduct the following steps.
    a. Set the parameter $a$ to be a *neighbor-based color estimate* value that is *close* to $p$ by using the colors of the neighboring pixels of $X$ that *have already been processed* (see discussion below).
    b. Perform alpha blending with $l$ over $a$ to get the parameter $b$ according to the formula $b = l \times \alpha + a \times (255 - \alpha)$ where $\alpha$ is the opacity of $Y$.
    c. Map $p$ to a new value $q = F_b^{-1}(F_a(p))$.
    d. Set the value of $Z$ to be $q$.
    e. Set the value of each remaining pixel in $W$, which is outside the region $P$, to be equal to that of the corresponding pixel in $I$.

For Step 3a above, there are several ways to determine the color estimate of a pixel using the colors of its neighbors that have already been processed, such as simply averaging the colors of the processed 4-neighbors of the pixel, or averaging those of the processed 8-neighbors with more weights on the horizontal and vertical members. We may also use more sophisticated techniques such as edge-directed prediction [105] for this purpose, as long as we use only processed pixels.

The reason for using *only* processed pixels is that these pixels are the ones that a receiver can reliably recover during watermark removal. This is to ensure that the same color estimates can be computed for lossless recovery. Specifically, the value $q$ of the first processed pixel is computed from the neighboring pixels outside the region $P$. Since the values of these pixels outside $P$ are unchanged, a receiver can therefore reliably recover the first pixel using a reverse mapping using $q$ and the values of neighboring pixels outside $P$. Each of the other unprocessed pixels is handled by using the processed pixels in a similar way.

To ensure that there always exists processed neighbors for accurate color estimates, we limit the pixels to be selected and processed next to be those with at least two already-processed neighbors in a four-pixel neighborhood. A consequence of this is that pixels around the outer edges of the watermark region are processed before those in the center. This can be clearly seen in Figure 8.5, where some of the intermediate outputs yielded during watermark embedding and removing are shown (the most obvious outer edges are seen in Figure 8.5(a)).

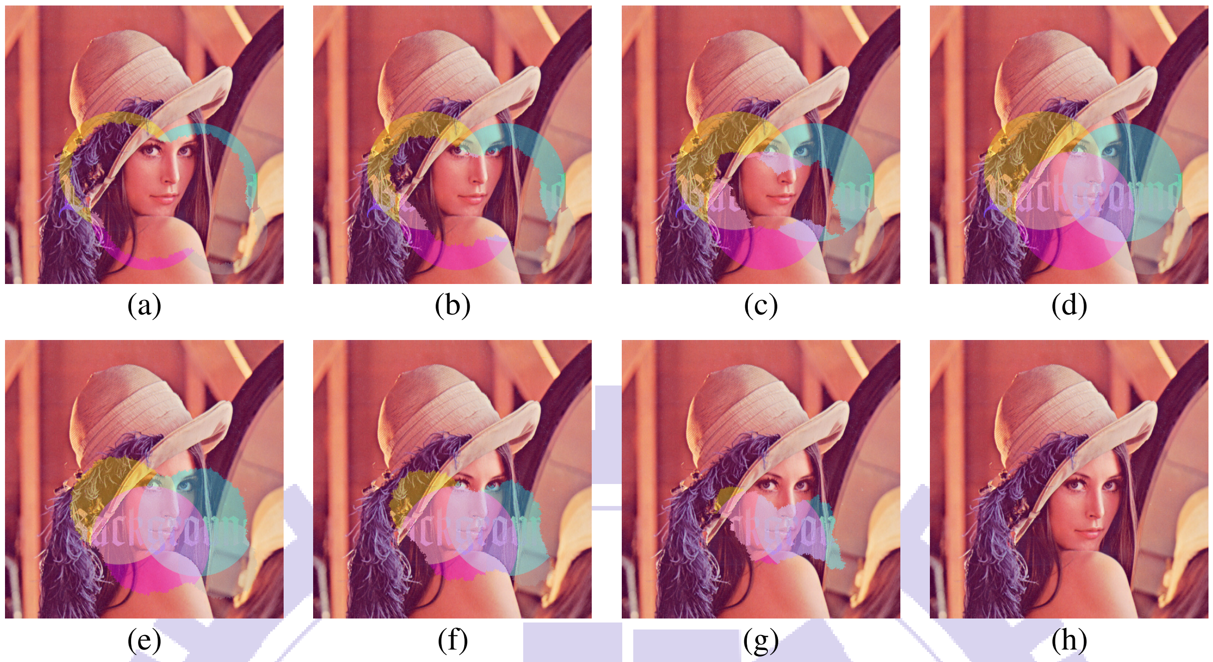


Figure 8.5. Illustration of pixel processing order in watermark embedding and removal. (a)-(d) Intermediate results of image watermarking when 25%, 50%, 75%, and 100% of the watermark pixels have been processed, respectively. (e)-(h) Intermediate results of image recovery when 25%, 50%, 75%, and 100% of the watermark pixels have been recovered, respectively.

## 8.5 Two-Fold Monotonically Increasing Compound Mapping

In Section 8.2, we mapped a pixel value to a preferred value by using a simple one-to-one function $F_x(p) = (p - x)_{\text{mod } 256}$. A problem of this mapping is that for certain values of $a$, $b$, and $p$, the mapped value will wrap around and deviate from the intended value. To solve this problem, we propose an alternative one-to-one function $F_x$ such that the compound mapping $q = F_b^{-1}(F_a(p))$ does not exhibit the wrap-around phenomenon. Specifically, the mapping always yields a value *close* to $b$ if $a$ and $p$ are *close* to each other for all values of $a$, $b$, and $p$. We will call this a *two-fold monotonically increasing* property, and will prove by a theorem that such a property holds if the one-to-one function $F_x$ has a *one-fold monotonically increasing* property. The definitions of both of these properties and the detail of the theorem are described in the following.

**Definition 8.1 (one-fold monotonically increasing one-to-one function).** A one-to-one function $F_a$ is one-fold monotonically increasing if for all values of $a$, $p_1$, and $p_2$, $F_a(p_1) < F_a(p_2)$ implies $|a - p_1| \leq |a - p_2|$.

**Lemma 8.3 (inverse monotonicity).** The inverse of a one-fold monotonically increasing function $F_x$ exhibits the following characteristic of *inverse monotonicity*:

for all values of $b$, $p_1'$, and $p_2'$, $p_1' < p_2'$ implies $|b - F_b^{-1}(p_1')| \leq |b - F_b^{-1}(p_2')|$.

***Proof.*** Let $p_1' = F_b(p_1)$ and $p_2' = F_b(p_2)$ for some $b$, $p_1$ and $p_2$. Then

$$|b - p_1| \leq |b - p_2|$$

by Definition 1. Also, we have $F_b^{-1}(p_1') = F_b^{-1}(F_b(p_1)) = p_1$, and $F_b^{-1}(p_2') = F_b^{-1}(F_b(p_2)) = p_2$, similarly. Substituting $p_1$ and $p_2$ into the above inequality, we get $|b - F_b^{-1}(p_1')| \leq |b - F_b^{-1}(p_2')|$. This completes the proof. □

**Definition 8.2 (two-fold monotonically increasing).** The compound mapping $q = F_b^{-1}(F_a(p))$ is two-fold monotonically increasing if for all values of $a$, $b$, $p_1$ and $p_2$, $|a - p_1| < |a - p_2|$ (i.e., if $a$ is closer to $p_1$ than $p_2$) implies $|b - q_1| \leq |b - q_2|$ (i.e., $b$ is at least as close to $q_1$ as $q_2$), where $q_1 = F_b^{-1}(F_a(p_1))$ and $q_2 = F_b^{-1}(F_a(p_2))$.

**Theorem 8.2 (two-fold monotonically increasing).** If $F_x$ is a one-fold monotonically increasing one-to-one function with a parameter $x$, then the compound mapping $q = F_b^{-1}(F_a(p))$ is two-fold monotonically increasing.

The proof of the above theorem can be found in Appendix B. We now show the existence of a one-fold monotonically increasing function $F_a(p)$ and how it works for any pixel value $a$ and $p$ in the range of 0 to 255, by way of an algorithm below.

**Algorithm 8.4: one-to-one mapping exhibiting one-fold monotonically increasing property.**

***Input***: a parameter $a$ and an input value $p$, each in the range of 0 to 255.

***Output***: a mapped output $p'$ in the range from 0 to 255.

***Steps:***

1. Initialize $p'$ to be zero.
2. Create a set $S$ with initial elements being the 256 values of 0 through 255.
3. Find a value $r$ in $S$ such that $|a - r|$ is the minimum, preferring a smaller $r$ in case of ties.
4. If $r$ is not equal to $p$, then remove $r$ from $S$, increment $p'$ by one, and go to Step 3; otherwise, take the final $p'$ as the output.

As an example, if we want to determine the function value $F_a(p)$ for $a = 3$ and $p = 1$ by the above algorithm, then we will find $r = 3$ in Step 3 of the above algorithm. But $r = 3 \neq 1 = p$, so 3 is removed from $S$ with $p'$ being incremented from 0 to 1. The subsequent iterations will compute $r$ to be 2, 4, and finally 1 which is equal to $p$, with the final value of $p'$ being taken to be 3 as the output.

The inverse of the one-to-one function described by Algorithm 8.4 is described below.

**Algorithm 8.5: inverse of the mapping function described by Algorithm 8.4.**

***Input*:** a parameter $b$ and an input value $p'$, each in the range of 0 to 255.

***Output*:** an output value $p$ that is in the range from 0 to 255.

***Steps:***

1. Create a set $S$ with the initial elements being the 256 values of 0 through 255.
2. Find a value $p$ in $S$ such that $|b - p|$ is the minimum, preferring a smaller $p$ in case of ties.
3. If $p'$ is larger than zero, then remove $p$ from $S$, decrement $p'$ by one, and go to Step 2; otherwise, take the final $p$ as the output.

As an example, if we want to compute $F_b^{-1}(p')$ for $b = 3$ and $p' = 3$ by the above algorithm, then we will find in Step 2 the sequence of 3, 2, 4, and 1 for the values of $p$, with $p'$ decreasing from 3, 2, 1 and then 0. The output is hence $p = 1$.

Note that in practice, we can pre-compute all 256×256 possible one-to-one mappings in both Algorithms 8.4 and 8.5 beforehand, so that the mapping $F_x$ and its inverse $F_x^{-1}$ can be implemented by efficient lookup-table operations of constant-time complexity. As proved by Theorem 8.2 and two extra lemmas (Lemmas 8.4 and 8.5) included in Appendix B, we can use the mapping and its inverse described in Algorithms 8.4 and 8.5, respectively, to map the pixel values of an image to the desired values of a watermarked image, such that the watermark is visually clear if $a$ is close to $p$. It is guaranteed that the original image can be recovered losslessly from the watermarked image, as proved by Theorem 8.1.

## 8.6 Experimental Results

A series of experiments implementing the proposed methods were conducted using the Java SE platform[17]. To quantitatively measure the effectiveness of the proposed method, we define a set of performance metrics here. First, the quality of a watermarked image *W* is measured by the peak signal-to-noise ratio (PSNR) of *W* with respect to the non-recoverable watermarked image *B* in the following way:

$$PSNR_W = 20 \times \log_{10}\left(255 \Big/ \sqrt{\frac{1}{w \times h}\sum_{y=1}^{h}\sum_{x=1}^{w}\left[W(x,y) - B(x,y)\right]^2}\right).$$

Also, the quality of a *recovered* image *R* is measured by the PSNR of *R* with respect to the original image *I* in a similar way:

$$PSNR_R = 20 \times \log_{10}\left(255 \Big/ \sqrt{\frac{1}{w \times h}\sum_{y=1}^{h}\sum_{x=1}^{w}\left[R(x,y) - I(x,y)\right]^2}\right).$$

It is desired to have the value of the $PSNR_W$ to be as high as possible, so that the watermarked image can be visually as close to the benchmark image as possible. For illicit recoveries, the $PSNR_R$ should be as low as possible to make the recovered image visually intolerable (e.g., very noisy). In particular, we want the region obscured by the watermark to be as noisy as possible in an illicitly recovered image. For this purpose, we introduce an additional quality metric for an illicitly recovered image that only takes into account the region *Q* covered by the watermark. Specifically, we measure the quality of the recovered image *R* by the following PSNR measure:

$$PSNR_Q = 20 \times \log_{10}\left(255 \Big/ \sqrt{\frac{1}{|Q|}\sum_{y=1}^{h}\sum_{x=1}^{w} SE_Q(x,y)}\right),$$

where

$$SE_Q(x,y) = \begin{cases} \left[R(x,y) - I(x,y)\right]^2 & \text{if } (x,y) \in Q; \\ 0 & \text{if } (x,y) \notin Q. \end{cases}$$

Six test images, each of dimensions 512×512, were used in the experiments. They are shown in Figure 8.6, referred to as "Lena," "baboon," "jet," "boat," "satellite," and "pepper," respectively, in the sequel. And seven test watermarks were used in the experiments as shown in Figure 8.7, hereinafter referred to as watermarks A, B, C, D, E, F, and G, respectively.

[17] The source code of the implementation can be downloaded at http://sites.google.com/site/ktyliu/lossless-visible-watermarking.

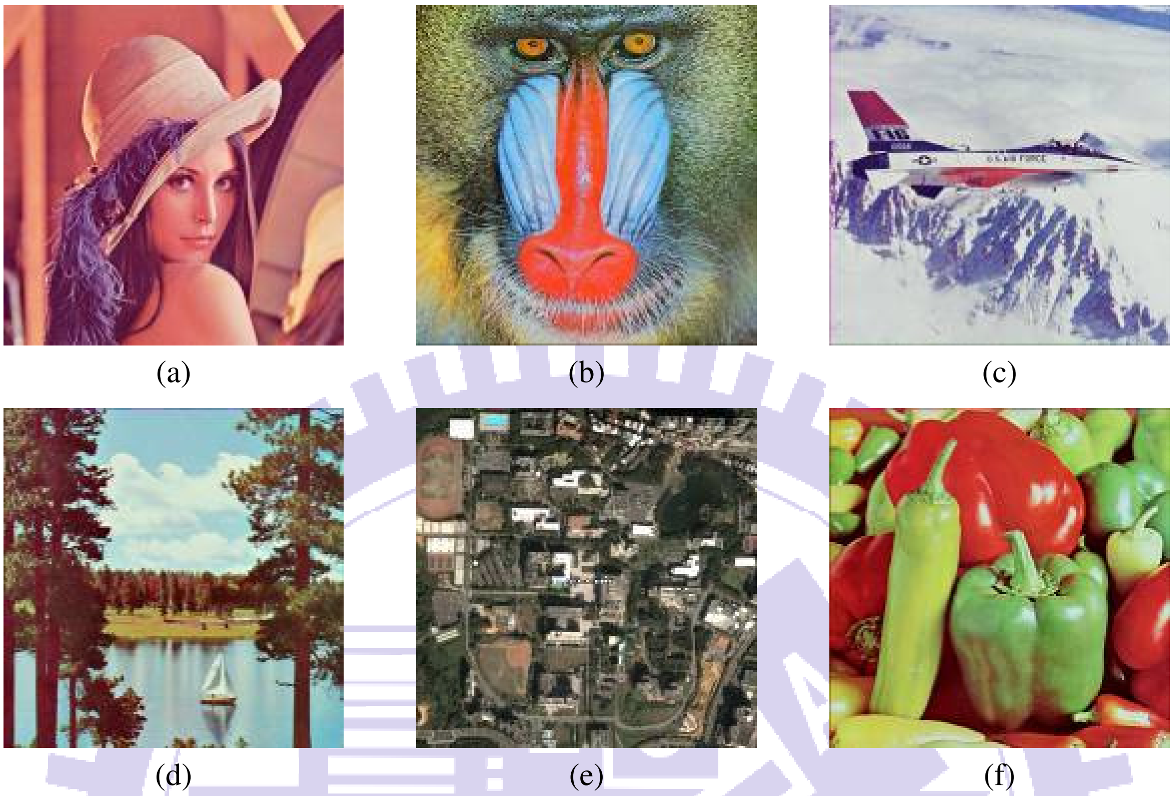


Figure 8.6. Test images used in experiments: (a) Lena; (b) Baboon; (c) Jet; (d) Sailboat; (e) A satellite image of NCTU campus; and (f) Pepper.

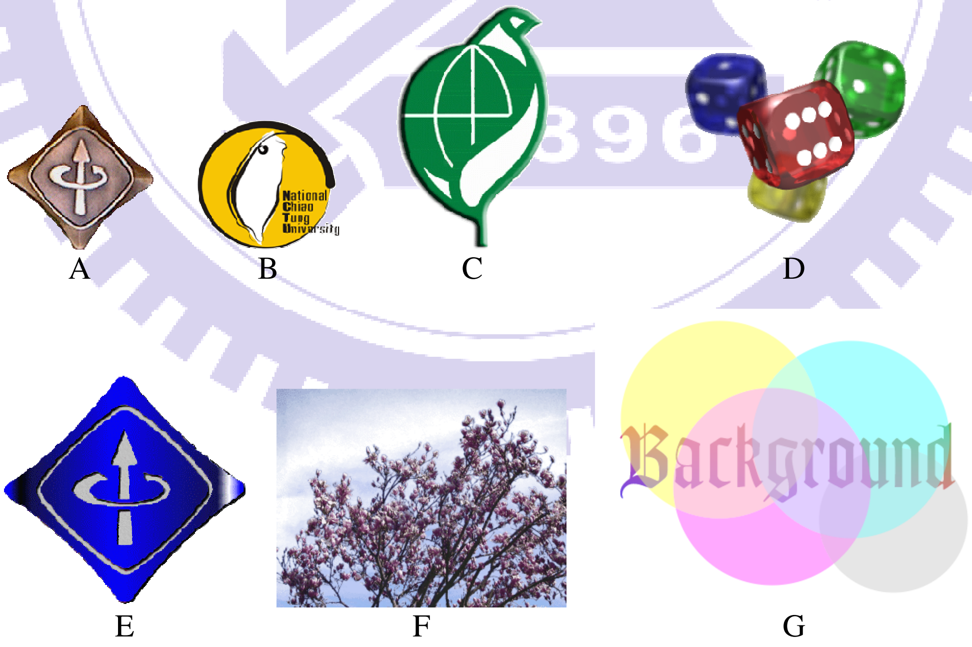


Figure 8.7. Watermarks A through G used in experiments.

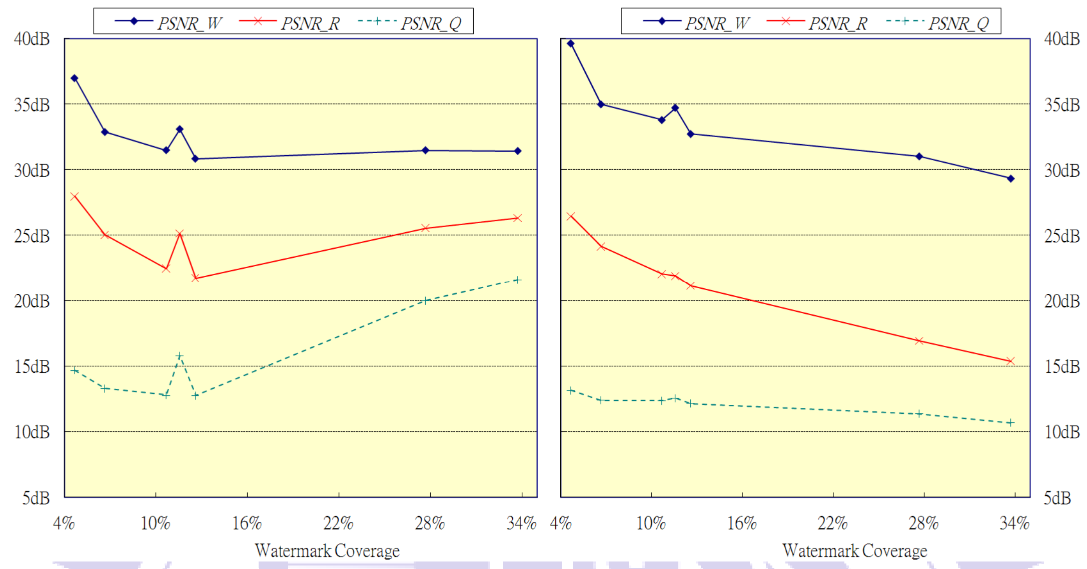
The width and height of each watermark are shown in Table 8.1, along with the number of non-transparent pixels in each watermark ($|P|$) and several other properties described next. The *average opacity*, as shown in the fourth column, is the average of the opacities of the pixels in the watermark, and the *coverage*, as shown in the last column, is the size of the watermark over the original image, which is computed as $|P|/(w \times h)$. The watermarks are listed in an increasing order of $|P|$, and the pixels in watermarks A, B, C, and E are either totally opaque or totally transparent, while watermarks D, F, and G contain semi-transparent pixels.

Table 8.1. Characteristics of watermarks A through G used in experiments.

| Watermark | Watermark Dimension | Non-transparent Pixels | Average Opacity | Watermark Coverage |
|---|---|---|---|---|
| A | 162×160 | 12,226 | 255 | 4.7% |
| B | 160×143 | 17,453 | 255 | 6.7% |
| C | 168×268 | 28,001 | 255 | 10.7% |
| D | 320×240 | 30,317 | 233 | 11.6% |
| E | 274×263 | 32,995 | 255 | 12.6% |
| F | 320×240 | 72,564 | 151 | 27.7% |
| G | 440×330 | 88,424 | 125 | 33.7% |

Each of the seven test watermarks was embedded in the six test images using the method described in Section 8.4 with the one-to-one compound mapping described in Section 8.5. The color estimate of a pixel was derived by averaging the available four-neighbors of that pixel. Such an experiment was conducted twice to test the effectiveness of the two proposed security protection measures: the mapping and the parameter randomization techniques. For the latter, both the parameters $a$ and $b$ of the compound mapping were adjusted randomly within a range of 25 with a uniform probability distribution.

A total of $7 \times 6 \times 2 = 84$ watermarked images were generated and for each watermarked image, recoveries using correct as well as incorrect keys were conducted. It was verified that the original images can be recovered *losslessly* from the watermarked ones for all the 84 test cases if correct keys were used. In Figure 8.8, we plot the average values of the $PSNR_W$ obtained after embedding a particular watermark in the six test images, as well as the average corresponding values of the $PSNR_R$ and $PSNR_Q$ obtained when incorrect keys were used for image recoveries.



(a) (b)

Figure 8.8. Average values of $PSNR_W$ obtained after watermark embedding and average values of $PSNR_R$ and $PSNR_Q$ obtained after illicit image recoveries. (a) Results yielded by parameter randomization. (b) Results yielded by mapping randomization.

Figure 8.9 shows three sets of the results, where Figures 8.9(c), 8.9(f), and 8.9(i) show the results where the parameter randomization technique was applied, while the other six images show the results where mapping randomization was applied. As can be seen from Figures 8.9(a) through 8.9(c), the watermarked images are visually close to the respective benchmark images, and the translucent color watermarks are distinctive in the watermarked images. There is some noise in the watermarking area of the watermarked images (yielded by large values of $|b - q|$) due to bad color estimations (with large values of $|a - p|$), which happen at edges in the images. The noise is scattered in the watermarking area when the mapping randomization technique was used, and coincides with the edges in the images when parameter randomization was applied.

The images recovered with correct keys are shown in Figures 8.9(d) through 8.9(f). As expected, the pixels of the recovered images are exactly identical to those of the original images. The robustness of the mapping randomization technique against illicit recoveries is evident as shown by the low $PSNR_Q$ in Figure 8.8. This comes from the fact that the incorrect recovery of one pixel value affects subsequent color estimations around that pixel. This *error avalanche* can be visually seen as patches of blurry noise in illicitly recovered images, as shown in Figures 8.9(g) through 8.9(h). On the other hand, the parameter randomization technique is weaker against illicit recoveries, especially in regions where the watermark has low opacity.

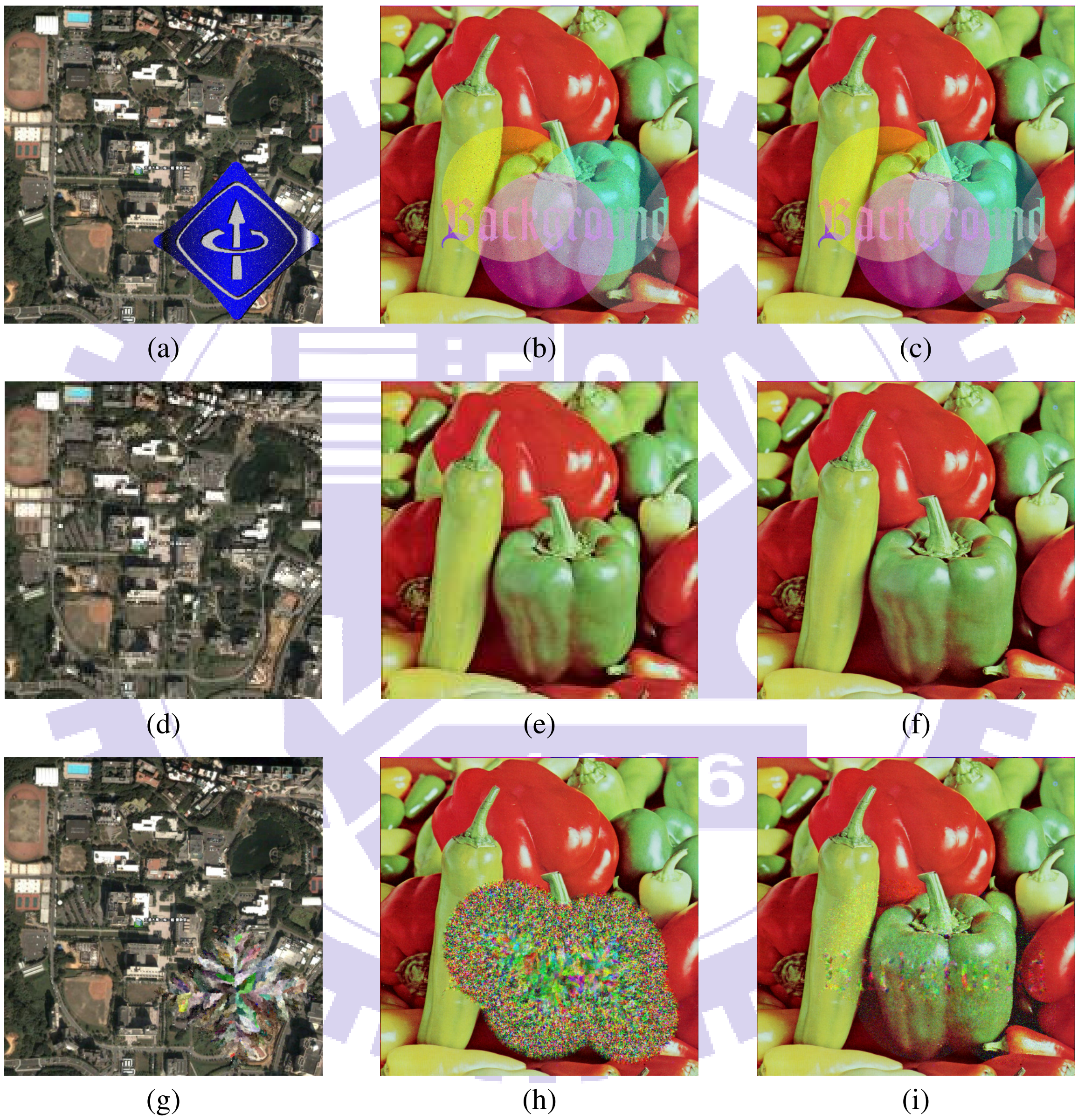


Figure 8.9. Watermarked images, licitly recovered images, and illicitly recovered images. (a)-(c) Watermarked images. (d)-(f) Licitly recovered images from images (a)-(c), respectively. (g)-(i) Illicitly recovered images from images (a)-(c), respectively.

A comparison of the capabilities of the proposed reversible visible watermarking method with those of four recently-published techniques is shown in Table 8.2. All but Hu [50] allows lossless recovery of the original image. Only Hu [50] and this study reported the *PSNR* for attempted recoveries using incorrect keys, and our results are better. In more detail, we embedded binary transparent watermarks similar to those used in Hu [50] using the proposed method, and obtained much better results (very low values of *PSNR* in the range of 12~14dB) than Hu's (37~39dB). More importantly, the proposed approach allows embedding of arbitrary-sized watermarks and has wider applicability than all four methods.

Table 8.2. Comparison of reversible visible watermarking techniques.

| Method | Legitimate recovery | Illegitimate recovery | Watermark size | Binary transparent watermark | Binary opaque watermark | Color translucent watermark |
|---|---|---|---|---|---|---|
| Hu [50] | 43~44 dB | 37~39dB | Unlimited | Yes | – | – |
| Hu [51] | Lossless | Not reported | Limited | Yes | – | – |
| Tsai [58] | Lossless | Not reported | Limited | Yes | – | – |
| Yip [59] | Lossless | Not reported | Unlimited | Yes | Yes | – |
| Proposed | Lossless | 12~14dB | Unlimited | Yes | Yes | Yes |

Finally, we conducted an experiment where a watermarked image produced using the proposed method was embedded into a Microsoft Word document, and it was verified that the lossless recoverability of the original image is unaffected by this embedding. Specifically, we embedded a visible watermark into a cover image to yield a watermarked image using the application written for the experiments; embedded the watermarked image into a Microsoft Word document; saved the Word document to disk and then reloaded the document later; saved the document in the Web page (html) format; located the saved image file (which is in the PNG file format) and ran watermark removal on the image file, which produced the original image losslessly. The experiment is repeated in a similar way using Microsoft PowerPoint, and it was verified that embedding the watermarked image into a Microsoft PowerPoint slide presentation does not affect the lossless recoverability of the image, provided that the image is not resized to a very small size.

## 8.7 Summary

In this chapter a new method for reversible visible watermarking with lossless image recovery capability has been proposed. The method uses one-to-one compound mappings that can map image pixel values to those of the desired visible watermarks. Relevant lemmas and theorems are described and proved to demonstrate the reversibility of the compound mappings for lossless reversible visible watermarking. The compound mappings allow different types of visible watermarks to be embedded, and two applications have been described for embedding opaque monochrome watermarks as well as translucent full-color ones. A translucent watermark is clearly visible and visually appealing, thus more appropriate than traditional transparent binary watermarks in terms of advertising effect and copyright declaration. The two-fold monotonically increasing property of compound mappings was defined and an implementation proposed that can provably allow mapped values to always be close to the desired watermark if color estimates are accurate. Also described are parameter randomization and mapping randomization techniques, which can prevent illicit recoveries of original images without correct input keys. Experimental results have demonstrated the feasibility of the proposed method and the effectiveness of the proposed security protection measures. Also, it is verified via experiments that the proposed method can be used for data hiding via office documents (using "data hiding in multimedia content"). Specifically, embedding a watermarked image into office documents does not affect the lossless recoverability of the original image.

# Chapter 9
# Conclusions and Suggestions for Future Works

In this dissertation, we investigated the problem of data hiding via office documents for various data hiding applications including covert communication, content authentication, copyright protection, and data association. We identified six areas of office documents where data can be hidden, five of which – texts, text formatting and layout, multimedia contents, multimedia formatting and layout, and auxiliary data – are logical areas for data hiding, where techniques devised for data hiding can in general be applied across multiple office document formats. Some unique characteristics of office documents compared to other media were described, and new methods for data hiding in office documents that takes into account these characters have been proposed.

In particular, a new approach to covert communication that leverages collaborative authoring was proposed. The data are hidden using the "data hiding via texts" approach in the experimental results, but it is also possible to hide the data in the other areas mentioned above as well under the proposed approach. By using change-tracking information, the degenerations to the content are more logical and hinder ignorant modification by skeptics compared to prior works that do not take advantage of the collaborative authoring characteristics. Future works can investigate into alternate degeneration databases and also using arithmetic coding instead of Huffman coding in data embedding for greater embedding capacity. Other data hiding methodologies based on disguising under collaborative efforts are open future research topics. In particular, data hiding in source code versioning using the CVS is an interesting future work.

The problem of quotation authentication is discussed in this dissertation in light of the frequent copying-and-pasting performed in office documents. The two-dimensional case is in particularly interesting as this is a new area that has not been studied before. The problem of quotation authentication is an under-studied area of research, and studies of techniques suitable for other popular formats such as e-mail messages and Internet web pages, for example, are possible future works. Another interesting future work is to consider the case where document authors sign their documents too, which will result in nested authenticable-quotations.

The challenge of the partial editing characteristics of office documents was also investigated in this dissertation study, and a weighted voting technique of expected sequence numbers was proposed. The technique was demonstrated to be able to successful identify sentences or slides that are copied from multiple sources for the source identification and the copyright protection applications. Adaptability of the proposed method for other data hiding applications can be pursued in future works. Other data hiding techniques appropriate for slides of presentations are also good future research topics.

It has also been demonstrated that the way of multimedia content embedding can be used for various data hiding applications where the structure of object groupings in a Microsoft Visio drawing was used for embedding data that can be used for applications such as drawing authentication and covert communication. It has been shown that the amount of data embeddable by the proposed method is on average twice the number of objects in the drawing. Alternate encodings that use more bits of 0's may be investigated in the future, as well as other variations of structures of object groupings which can provide larger data embedding capacities.

Finally, it was demonstrated that hiding data in a multimedia content and then embedding the stego-media into an office document is a viable way for data hiding via office documents. Specifically, we can embed a visible watermark into an image and then embed the watermarked image into a Microsoft Word or PowerPoint document. The watermarked image can be extracted from the stego-document, and it is demonstrated that the original image can be recovered losslessly from the extracted image.

Table 9.1 shows a summary of the proposed methods described in Chapter 3 through Chapter 8, including the data hiding applications that were described, the office document types that were used, and the areas for data hiding leveraged. It is noted that some of the proposed methods may be extended to allow for other data hiding applications or office document types. The table only lists those that are explicitly discussed or mentioned in the chapters.

As mentioned in the motivation of this study, there are still few techniques proposed that address the problem of data hiding via office documents. This research area is of great theoretical as well as practical importance considering the flexibility and ubiquity of office documents. It is expected that there will be new techniques proposed in the future for data hiding in all of the six areas for researches described in Chapter 2. New approaches to data hiding and new data hiding applications that take into consideration the properties of office documents are also exciting future research topics.

Table 9.1. Summary of proposed data hiding methods in this dissertation study.

| | |
|---|---|
| *Data Hiding Applications* | |
| Covert communication | Chapter 3, Chapter 7 |
| Copyright protection | Chapter 8 |
| Data association | Chapter 4, Chapter 5, Chapter 6 |
| Media authentication | Chapter 4, Chapter 5, Chapter 7 |
| *Office Document Types* | |
| Word processing documents | Chapter 3, Chapter 4, Chapter 8 |
| Spreadsheet documents | Chapter 5 |
| Slide presentation documents | Chapter 6, Chapter 8 |
| Vector graphics documents | Chapter 7 |
| *Data Hiding Research Directions in Office Documents* | |
| Data hiding via texts | Chapter 3 |
| Data hiding via text formatting and layout | Chapter 4, Chapter 6 |
| Data hiding via multimedia contents | Chapter 8 |
| Data hiding via multimedia formatting and layout | Chapter 7 |
| Data hiding via auxiliary data | Chapter 3, Chapter 4, Chapter 5 |

# Appendix A

# Programmatic Manipulation of Office Documents

As mentioned in the introductory chapter, office documents are very versatile and their document format specifications can be extremely complicated. To ease the manipulation of office documents and hence extend the capabilities of existing office software applications in user-defined ways, many of the office applications allow documents to be manipulated *programmatically*. Such a technique is often called *Automation* [71], and involves the calling or embedding of the relevant office software components in a user's program.

The office software application typically exposes and documents a set of public APIs (Application Programming Interface) that a caller can use to open, read, or manipulate office documents. The automation can be performed *inside* an office software application, in which case the user's program is often called a *macro*, or *outside* the application using inter-process communication techniques such as COM (Component Object Model).

As an example, the code excerpt in Figure A.1 demonstrates the manipulation of a Microsoft Word document from an external program using the C$^{\#}$ programming language and the Microsoft Office .NET APIs (Microsoft.Office.Interop.Word). Specifically, when an object of the class "Word.ApplicationClass" is created, the Microsoft Word application is essentially launched, albeit hidden and not visible to the user initially (it is possible to set the "Visible" property of the object to true to make the application visible, in which case the application will appear no different to that launched manually by a user).

Any method invocations on the object (named "wordApp" in the figure) essentially results in inter-process calls that command the Word application to execute the desired functionality. In this example, the Word application is commanded to open a certain existing document; all existing tracked changes are accepted; the number of sentences contained in the document is queried; and selected sentences in the document are degenerated and revised (the excerpt is taken from a prototype that implements the method proposed in Chapter 3).

Using such an Automation technique allows different office documents to be manipulated easily, allowing us to focus on the algorithms and approaches of effective data hiding techniques specific to office documents rather than delving into the details of office document formats.

```
Using Word = Microsoft.Office.Interop.Word;
...
Word.Application wordApp = new Word.ApplicationClass();
Word.Document doc = wordApp.Documents.Open(file, …);
doc.Revisions.AcceptAll();
int sentences = doc.Sentences.Count;
int[] permutation = GetPermutation(sentences);
for (int i = 0; i < sentences && !msgEncoder.EOF(); ++i) {
  Word.Range sentence = doc.Sentences[permutation[i]];
  string d = sentence.Text;
  ...
  string d_ = degenerator.Degenerate(d, msgEncoder);
  if (d_ == null) continue;
  // Degenerate
  doc.TrackRevisions = false;
  DegenerateLevenshtein(sentence, d_);
  // Revert
  doc.TrackRevisions = true;
  DegenerateLevenshtein(doc.Sentences[permutation[i]], d);
}
```

Figure A.1. Code excerpt demonstrating the manipulation of a Microsoft Word document programmatically.

# Appendix B

# Proofs of Theorems and Lemmas in Section 8.5

**Theorem 8.2 (two-fold monotonically increasing).** If $F_x$ is a one-fold monotonically increasing one-to-one function with a parameter $x$, then the compound mapping $q = F_b^{-1}(F_a(p))$ is two-fold monotonically increasing.

***Proof.*** In the beginning, we prove that for all values of $a$, $p_1$, and $p_2$, $F_a(p_1) < F_a(p_2)$ if $|a - p_1| < |a - p_2|$ by showing that both the inequality (i) $F_a(p_1) > F_a(p_2)$ and the equality (ii) $F_a(p_1) = F_a(p_2)$ are impossible if $|a - p_1| < |a - p_2|$. First, (i) is impossible by the definition of one-fold monotonically increasing function (Definition 1), since if not so, it will then imply that $|a - p_2| \le |a - p_1|$, which is a contradiction. Next, (ii) is also impossible because $F_x$ is a one-to-one function, implying $p_1 = p_2$, which contradicts the condition $|a - p_1| < |a - p_2|$. This completes the first part of the proof that

$$F_a(p_1) < F_a(p_2) \text{ if } |a - p_1| < |a - p_2|.$$

In the second part of proof, by regarding $F_a(p_1)$ as $p_1'$ and $F_a(p_2)$ as $p_2'$, and substituting them into the inequalities of Lemma 3, we reach the fact that for all values of $a$, $b$, $p_1$ and $p_2$,

$$|b - F_b^{-1}(F_a(p_1))| \le |b - F_b^{-1}(F_a(p_2))| \text{ if } F_a(p_1) < F_a(p_2).$$

Combining the results of the two parts of proof above, we have

$$|b - F_b^{-1}(F_a(p_1))| \le |b - F_b^{-1}(F_a(p_2))| \text{ if } |a - p_1| < |a - p_2|,$$

or equivalently,

$$|b - q_1| \le |b - q_2| \text{ if } |a - p_1| < |a - p_2|,$$

where $q_1 = F_b^{-1}(F_a(p_1))$, and $q_2 = F_b^{-1}(F_a(p_2))$. That is, the two-fold monotonically increasing property holds. This completes the proof. □

**Lemma 8.4** The function described by Algorithm 8.4 is one-to-one and one-fold monotonically increasing.

***Proof.*** In Step 4 of Algorithm 8.4, we always remove a unique element from the set $S$ and in turn increment $p'$, and so each of the 256 possible input values of $p$ will yield its own unique output $p'$ value. Thus Algorithm 8.4 indeed describes a one-to-one function for all values of $a$.

Furthermore, since we remove values of $r$ from $S$ in an increasing order of $|a - r|$, a larger value of $p'$ means that $r$ is farther away from $a$. This means that the value of $p' = F_a(p)$ yielded by Algorithm 8.4 satisfies the one-fold monotonically increasing property: $F_a(p) < F_a(p')$ implies $|a - p| \le |a - p'|$. □

**Lemma 8.5** The function described in Algorithm 8.5 is the inverse of the function described in Algorithm 8.4.

***Proof.*** If we set the value of input $b$ in Algorithm 8.5 to be the input $a$ in Algorithm 8.4, then the set $S$ in Algorithms 8.4 and 8.5 will always contain exactly the same elements for each iteration. This is because in each iteration the value $r$ picked by Step 3 of Algorithm 8.4 will be the same as the value $p$ picked by Step 2 of Algorithm 8.5, and this same value is removed respectively in Step 4 of Algorithm 8.4 and Step 3 of Algorithm 8.5.

For all values of input $p'$, Algorithm 8.5 will pick the $(p' + 1)$-th item in the sequence of $p$'s computed in Step 2 as the final output $p$, which we denote as $p^*$. Since the sequence of $p$'s selected in Step 2 of Algorithm 8.5 is exactly identical to the sequence of $r$'s picked in Step 3 of Algorithm 8.4, if the value $p^*$ is used as the input $p$ to Algorithm 8.4, then $r$ will match $p^*$ exactly after $(p' + 1)$ iterations. Algorithm 8.4 will therefore output the same $p'$ value, demonstrating that the function it described is the inverse of that described by Algorithm 8.5. Since the functions described by the two algorithms are one-to-one, the function described by Algorithm 8.5 is the inverse of that described by Algorithm 8.4. This completes the proof. □

# References

[1] P. Moulin and R. Koetter, “Data-hiding codes,” *Proceedings of the IEEE*, vol. 93, no. 12, pp. 2083-2126, 2005.

[2] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, “Techniques for data hiding,” *IBM Systems Journal*, vol. 35, no. 3-4, pp. 313-336, 1996.

[3] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, “Information hiding — a survey,” *Proceedings of the IEEE*,vol. 87, no. 7, pp. 1062-1078, July 1999.

[4] I. J. Cox, and M. L. Miller, “The first 50 years of electronic watermarking,” *Journal of Applied Signal Processing*, 2002, 2, pp. 126-132.

[5] N. F. Johnson, Z. Duric, and S. Jajodia, *Information hiding: steganography and watermarking - attacks and countermeasures*, Kluwer Academic Publishers, Boston, 2001.

[6] R. Chandramouli, M. Kharrazi, and N. Memon, “Image steganography and steganalysis concepts and practice,” *Digital Watermarking Lecture Notes in Computer Science*, vol. 2939, pp. 35-49, 2004.

[7] G. Cantrell and D. D. Dampier , “Experiments in hiding data inside the file structure of common office documents: a stegonography application,” *Proceedings of the 2004 International Symposium on Information and Communication Technologies*, Las Vegas, Nevada, pp. 146-151, 2004.

[8] ISO/IEC 29500:2008, *Information technology -- Document description and processing languages -- Office Open XML File Formats.*

[9] 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, no. 5, pp. 767-782, 2001.

[10] J. J. K. O' Ruanaidh and T. Pun, “Rotation, scale and translation invariant digital image watermarking,” in *Proceedings of IEEE International Conference on Image Processing*, Santa Barbara, CA USA, vol. 1, pp. 536-539, October 1997.

[11] K. Bennett, “Linguistic steganography: survey, analysis, and robustness concerns for hiding information in text,” *CERIAS Tech Report* 2004-13, Center for Education and Research in Information Assurance and Security, Purdue University, West Lafayette, May 2004.

[12] P. Wayner, “Mimic functions,” *Cryptologia, XVI(3)*, pp.193-214, 1992.

[13] P. Wayner, *Disappearing Cryptography: Information Hiding: Steganography and Watermarking*, Morgan Kaufmann, 2nd edition, pp. 81-128, 2002.

[14] K. Maher, Texto, circulating on the web, February 1995.

[15] M. Chapman, I. D. George, and R. Marc, "A practical and effective approach to large-scale automated linguistic steganography," in *Proceedings of the Information Security Conference (ISC '01)*, Malaga, Spain, pp. 156-165, October 2001.

[16] "Spam Mimic," online at http://www.spammimic.com (accessed: March 12, 2010).

[17] I. A. Bolshakov, "A method of linguistic steganography based on collocationally-verified synonymy," in *Proceedings of the 6th Information Hiding Workshop (IH 2004)*, Toronto, Ontario, Canada, pp. 180-191, May 2004.

[18] H. M. Meral, E. Sevinc, E. Unkar, B. Sankur, A. S. Ozsoy, and T. Gungor, "Syntactic tools for text watermarking," in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents IX*, San Jose, CA, January-February 2007.

[19] M. Topkara, C. Taskiran, and E. J. Delp, "Natural Language Text Watermarking," in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents VI*, San Jose, CA, January 2005.

[20] C. Grothoff, K. Grothoff, L. Alkhutova, R. Stutsman, and M. Atallah "Translation-based steganography," in *Proceedings of Information Hiding Workshop (IH 2005)*, pp. 213-233, 2005

[21] R. Stutsman, C. Grothoff, M. Attallah, and K. Grothoff, "Lost in just the translation," in *Proceedings of the 2006 ACM symposium on Applied computing*, pp. 338-345, 2006.

[22] M. Topkara, U. Topkara, and M. J. Atallah, "Information hiding through errors: a confusing approach," in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents IX*, San Jose, CA, January-February 2007.

[23] I. S. Lee and W. H. Tsai, "Data hiding in emails and applications by unused ASCII control codes," *Journal of Information Technology and Applications*, vol. 3, no. 1, pp. 13-24, September 2008.

[24] I. S. Lee and W. H. Tsai, "Security protection of software programs by information sharing and authentication techniques using invisible ASCII control codes," *International Journal of Network Security*, vol. 10, no. 1, pp. 1-10, January 2010.

[25] A. E. Ali, "A new text steganography method by using non-printing unicode characters," Eng. & Tech. Journal, vol.28, no.1, 2010.

[26] I. S. Lee and W. H. Tsai, “A new approach to covert communication via PDF Files,” *Signal Processing*, vol. 90, no. 2, pp. 557-565, February 2010.

[27] F. J. Mabry, J. R. James, and A. J. Ferguson, “Unicode steganographic exploits: maintaining enterprise border security,” *IEEE Security and Privacy,* vol. 5, no. 5, 32-39, 2007.

[28] N. F. Johnson and S. Jajodia, “Steganalysis: The Investigation of Hidden Information,” in *Proceedings of the IEEE Information Technology Conference*, Syracuse, New York, USA, pp. 113-116, September 1998.

[29] M. Taskiran, U. Topkara, M. Topkara, and E. Delp, “Attacks on lexical natural language steganography systems,” in *Proceedings of SPIE*, vol. 6072, pp. 97-105, 2006.

[30] Z. S. Yu, L. S. Huang, Z. L. Chen, L. J. Li, X. X. Zhao, and Y. W. Zhu, “Steganalysis of synonym-substitution based natural language watermarking,” *International Journal of Multimedia and Ubiquitous Engineering*, vol. 4, pp. 21-34, 2009.

[31] G. Simmons, “The prisoners’ problem and the subliminal channel,” in *Advances in Cryptology: Proceedings of Crypto*, vol. 83, pp. 51-67, 1984.

[32] N. Maxemchuk and S. Low, “Marking text documents,” in *International Conference of Image Processing*, vol. 3, pp. 13-17, 1997.

[33] J. T. Brassil, and N. F. Maxemchuk, “Copyright protection for the electronic distribution of text Documents,” *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1181-1196, July 1999.

[34] S. Zhong, X. Cheng, and T. Chen, “Data hiding in a kind of PDF texts for secret communication,” *International Journal of Network Security*, vol. 4, pp. 17-26, 2007.

[35] R. Villán, S. Voloshynovskiy, O. Koval, J. E. Vila-Forcén, E. Topak, F. Deguillaume, Y. Rytsar, and T. Pun, “Text data-hiding for digital and printed documents: theoretical and practical considerations,” in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents VIII*, vol. 6072, 2006.

[36] S. Walton, “Image authentication for a slippery new age,” *Dr. Dobb's Journal-Software Tools for the Professional Programmer*, vol. 20, no. 4, pp. 18-27, 1995.

[37] Q. Mei, E. Wong, and N. Memon, “Data hiding in binary text documents,” in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents III*, vol. 4314, pp. 369–375, August 2001.

[38] M. Wu and B. Liu, “Data hiding in binary image for authentication and annotation,” *IEEE Transactions on multimedia*, vol. 6, no. 4, pp. 528-538, 2004.

[39] C. H. Tzeng and W. H. Tsai, "A new approach to authentication of binary images for multimedia communication with distortion reduction and security enhancement," *IEEE Communication Letters*, vol. 7, no. 9, pp. 443–445, September 2003.

[40] D. C. Wu, and W. H. Tsai, "A steganographic method for images by pixel-value differencing," *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1613-1626, 2003.

[41] P. C. Su and C. C. Kuo, "Steganography in JPEG2000 compressed images," *IEEE Transactions on Consumer Electronics*, 49, 4, pp. 824–832, 2003.

[42] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1673−1687, June 1997.

[43] M. Swanson, M. Kobayashi, and A. Tewfik, "Multimedia data-embedding and watermarking technologies," *Proceedings of the IEEE*, vol. 86, pp. 1064-1088, 1998.

[44] M. S. Kankanhalli, Rajmohan, and K. R. Ramakrishnan, "Adaptive visible watermarking of images," in *IEEE International Conference on Multimedia Computing and Systems*, vol. 1, pp. 568–573, 1999.

[45] Y. Hu, and S. Kwong, "Wavelet domain adaptive visible watermarking," *Electronics Letters*, vol. 37, no. 20, pp. 1219–1220, September 2001.

[46] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kankanhalli, "A DCT domain visible watermarking technique for images," in *IEEE International Conference in Multimedia and Expo*, vol. 2, pp. 1029–1032, Jul. 2000.

[47] G. Braudaway, K. A. Magerlein, and F. Mintzer, "Protecting publicly available images with a visible image watermark," in *Proceedings of the SPIE International Conference on Electronic Imaging*, vol. 2659, pp. 126–133, February 1996.

[48] C. Lu, S. Huang, C. Sze, and H. Liao, "Cocktail watermarking for digital image protection," *IEEE Transactions on Multimedia*, vol. 2, no. 4, pp. 209-224, 2000.

[49] Y. J. Cheng, and W. H. Tsai, "A new method for copyright and integrity protection for bitmap images by removable visible watermarks and irremovable invisible watermarks," presented at the *2002 International Computer Symposium – Workshop on Cryptology and Information Security*, Hualien, Taiwan, R.O.C., December 2002.

[50] Y. Hu, S. Kwong, and J. Huang, "An algorithm for removable visible watermarking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 1, pp.129–133, January 2006.

[51] Y. Hu and B. Jeon, “Reversible visible watermarking and lossless recovery of original images,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 11, pp. 1423–1429, November 2006.

[52] M. Awrangjeb, and M. S. Kankanhalli, “Lossless watermarking considering the human visual system,” in *Proceedings of the International Workshop on Digital Watermarking (IWDW 2003)*, Seoul, Korea, October 2003.

[53] M. Awrangjeb, and M. S. Kankanhalli, “Reversible watermarking using a perceptual model,” *Journal of Electronic Imaging*, vol. 14, no. 013014, March 2005.

[54] C. Chang, P. Pai, C. Yeh, and Y. Chan, “A high payload frequency-based reversible image hiding method,” *Information Sciences*, vol. 180, no. 11, pp. 2286-2298, June 2010.

[55] C. De Vleeschouwer, J. F. Delaigle, and B. Macq, “Circular interpretation of bijective transformations in lossless watermarking for media asset management,” *IEEE Transactions on Multimedia*, vol. 5, no. 1, pp.97–105, March 2003.

[56] J. Tian, “Reversible data embedding using a difference expansion,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp.890–896, August 2003.

[57] C. Tsai, H. Chiang, K. Fan, and C. Chung, “Reversible data hiding and lossless reconstruction of binary images using pair-wise logical computation mechanism,” *Pattern Recognition*, vol. 38, no. 11, pp. 1993-2006, 2005.

[58] H. M. Tsai and L. W. Chang, “A high secure reversible visible watermarking scheme,” in *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME2007)*, Beijing, China, pp. 2106–2109, July 2007.

[59] S. K. Yip, O. C. Au, C. W. Ho, and H. M. Wong, “Lossless visible watermarking,” in *IEEE International Conference on Multimedia & Expo*, pp.853–856, July 2006.

[60] X. Niu, C. Shao, and X. Wang, “A survey of digital vector map watermarking,” *International Journal of Innovative Computing, Information and Control*, vol. 2, no. 6, pp. 1301–1316, 2006.

[61] K. R. Kwon, B. J. Jang, E. J. Lee, and Y. Huh, “Copyright protection of architectural CAD drawing using the multiple watermarking scheme,” in *Proceedings of the International Conference on Multimedia & Expo (ICME 2004)*, pp. 871–874, June 2004.

[62] V. Solachidis and I. Pitas, “Watermarking polygonal lines using fourier descriptors,” *IEEE Computer Graphics and Applications*, vol. 24, no. 3, pp. 44–51, 2004.

[63] V. R. Doncel and N. Nikolaidis and I. Pitas, “An optimal detector structure for the Fourier descriptors domain watermarking of 2D vector graphics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 5, pp. 851–863, 2007.

[64] D. H. Im and H. Y. Lee, S. J. Ryu and H. K. Lee, “Vector watermarking robust to both global and local geometrical distortions,” *IEEE Signal Processing Letters*, vol. 15, pp. 789–792, 2008.

[65] A. Giannoula, N. Nikolaidis, and I. Pitas, “Watermarking of sets of polygonal lines using fusion techniques,” in *Proceedings of the International Conference on Multimedia & Expo (ICME2002)*, Laussane, Switzerland, pp. 26–29, August 2002.

[66] R. Ohbuchi, “A shape-preserving data embedding algorithm for NURBS curves and surfaces,” in *Proceedings of the Computer Graphics International*, Canmore, Canada, pp. 177–180, 1999.

[67] W. Yang and L. Chen, “A novel steganography method via various animation effects in PowerPoint files,” in *International Conference on Machine Learning and Cybernetics*, Kunming, China, pp. 3102-3107, July 2008.

[68] M. Q. Jing, W. C. Yang and L. H. Chen, 2009, "A new steganography method via various animation timing effects in Powerpoint files," in *International Conference on Machine Learning and Cybernetics*, Baoding China, pp. 2840-2845, July 2009.

[69] A. Castiglione, A. De Santis, and C. Soriente, “Taking advantages of a disadvantage: Digital forensics and steganography using document metadata,” *The Journal of Systems & Software*, vol. 80, no. 5, pp. 750-764, 2007.

[70] Y. Liu, X. Sun, Y. Liu, and C. Li, “MIMIC-PPT: Mimicking-based steganography for Microsoft PowerPoint document,” *Information Technology Journal*, vol. 7, no. 4, pp. 654-660, 2008.

[71] *Microsoft Office 97 Visual Basic Programmer's Guide* (Microsoft Professional Editions Series), 1997.

[72] S. Page, T. Johnsgard, U. Albert, and C. Allen, “User customization of a word processor,” in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems: Common Ground*, p. 346, 1996.

[73] H. Yu, P. Yin, S. Cheng, X. Kong, A. Gelman, and R. Fish, “Smart Media: empower media with active data hiding,” *Lecture Notes in Computer Science*, pp. 5-16, 2001.

[74] Microsoft, “Remove personal or hidden information,” online at http://office.microsoft.com/en-us/word/HP051901021033.aspx (accessed: March 15, 2010).

[75] Microsoft, "The remove hidden data tool for Office 2003 and Office XP," online at http://support.microsoft.com/kb/834427 (accessed: March 15, 2010).

[76] H. Kwon, Y. Kim, S. Lee, and J. Lim, "A tool for the detection of hidden data in Microsoft compound document file format," in *Proceedings of 2008 International Conference on Information Science and Security (ICISS 2008)*, Seoul, Korea, pp. 141-146, January, 2009.

[77] Y. Liu, X. Sun, Y. Liu, and R. Xiao, "File-update based steganography for Microsoft PowerPoint files," in *Proceedings of the 2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 11-15, August 2008.

[78] J. Park, B. Park, S. Lee, S. Hong, and J. Park, "Extraction of residual information in the Microsoft PowerPoint file from the viewpoint of digital forensics considering PerCom environment," in *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, pp. 584-589, Washington, DC, USA, 2008.

[79] B. Park, J. Park, S. Lee, "Data concealment and detection in Microsoft Office 2007 files," *Digital Investigation*, vol. 5, no. 3-4, pp.104-114, March 2009.

[80] S. Inoue, K. Makino, I. Murase, O. Takizawa, T. Matsumoto, and H. Nakagawa, "A proposal on information hiding methods using XML," in *Proceedings of the 1st NLP and XML Workshop*, pp. 55-62, November 2001.

[81] Y. H. Chang and W. H. Tsai, "Steganographic method for copyright protection of HTML documents", in *Proceedings of 2003 National Computer Symposium*, Taichung, Taiwan, December 2003.

[82] I. S. Lee and W. H. Tsai, "Secret communication through web pages using special codes in HTML files," *International Journal of Applied Science and Engineering*, vol. 6, no. 2, November 2008, pp. 141-149.

[83] X. G. Sui and H. Luo, "A new steganography method based on hypertext," in *Proceedings of the Radio Science Conference*, pp. 181-184, August 2004.

[84] Advanced Encryption Standard (AES), FIPS Pub 197, November 2001, online at http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf (accessed: January 11, 2010)

[85] P. Brian, Common Errors in English, online at http://www.wsu.edu/~brians/errors/ (accessed: 25 August, 2006).

[86] Princeton University, WordNet v2.1, a lexical database for the English language, 2005.

[87] Google, Google SOAP Search API (beta), online at: http://www.google.com/apis/.

[88] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN systems*, vol. 30, pp. 107-117, 1998.

[89] C. Fernstrom, "Management of trusted citations," in *Proceedings of 2003 ACM Symposium on Document Engineering*, Grenoble, France, pp. 243-245, 2003.

[90] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Advances in Cryptology - Crypto 96 Proceedings, Lecture Notes in Computer Science*, 1109, Springer-Verlag, 1996.

[91] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: keyed-hashing for message authentication," *Internet RFC 2104*, February 1997.

[92] W. Stallings, *Cryptography and Network Security, Principles and Practice, 2nd ed.*, Prentice-Hall International, 1999.

[93] PKCS #1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 2002.

[94] S. Josefsson, "RFC 4398 - Storing certificates in the domain name system (DNS)," March 2006, online at http://www.ietf.org/rfc/rfc4398.txt (accessed: March 19, 2010).

[95] R. Gennaro and P. Rohatgi, "How to sign digital streams," *Information and Computation archive*, vol. 165, no. 1, pp. 100–116, 2001.

[96] A. Perrig, R. Canetti, D. Song, and J.D. Tygar, "Efficient and secure source authentication for multicast," in *Proceedings of Network and Distributed System Security Symposium*, pp. 35–46, 2001.

[97] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," *Lecture Notes in Computer Science*, vol. 3621, pp. 17–36, 2005.

[98] F. C. Rice, "Building a COM Add-in for Microsoft Office XP Using Microsoft Visual Basic 6.0," *Microsoft Office XP Technical Articles*, Microsoft Corporation, 2002.

[99] Harlan Carvey, *Windows Forensics and Incident Recovery*, Addison-Wesley Professional, 2004.

[100] M. Muterspaugh, H. Liu, and W. Gao, "Thomson proposal outline for WRAN," proposal for IEEE P802.22 Wireless RANs, doc.: IEEE802.22-05/0096r1, 2005.

[101] S. S. Tseng and W. H. Tsai, "High confidence information systems mid-term report," NSC Advanced Technologies and Applications for Next Generation Information Networks, October 2002.

[102] Xilinx, "Digital Filtering," DSP Design Flow, 2003, online at http://users.ece.gatech.edu/~hamblen/4006/xup/dsp_flow/slides/ (accessed: March 19, 2010).

[103] Microsoft Corporation, Visio 2003 SDK Documentation, MSDN Library, Office Developer Center, Visio 2003, online at http://msdn.microsoft.com/en-us/library/bb421578(office.11).aspx (accessed: January 2, 2010).

[104] A. Lumini and D. Maio, "Adaptive positioning of a visible watermark in a digital image," in *Proceedings of the International Conference on Multimedia & Expo (ICME2004)*, Taipei, Taiwan, R.O.C., pp.967–970, June 2004.

[105] X. Li, and M. T. Orchard, "Edge-directed prediction for lossless compression of natural images," *IEEE Transactions on Image Processing*, vol. 10, no. 6, pp. 813–817, June 2001.

# Vitae

Tsung-Yuan Liu was born in Taipei, Taiwan on September 15, 1977, and studied in Johannesburg, South Africa from 1987 to 1998. He received the B.S. degree in electrical engineering from the University of the Witwatersrand, Johannesburg, South Africa, the M.B.A. degree from National Taiwan University, Taipei, Taiwan, R.O.C., and is studying towards the Ph.D. degree at the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, R. O. C.

He is currently a Software Engineer at Google, Taipei, Taiwan. His research interests include data hiding, image processing, web search, data mining, and artificial intelligence.

# List of Publications

1. Tsung-Yuan Liu and Wen-Hsiang Tsai, “A new steganographic method for data hiding in microsoft word documents by a change tracking technique,” *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 1, March 2007, pp. 24-30 (based on the content of Chapter 3).
2. Tsung-Yuan Liu and Wen-Hsiang Tsai, “Generic lossless visible watermarking -- A new approach,” *IEEE Transactions on Image Processing*, vol. 19, no. 5, pp. 1224-1235, May 2010 (based on the content of Chapter 8).
3. Tsung-Yuan Liu and Wen-Hsiang Tsai, “Robust watermarking in slides of presentations by blank space coloring: a new approach,” *LNCS Transactions on Data Hiding and Multimedia Security IV*, Lecture Notes in Computer Science (LNCS), vol. 5510, Y. Q. Shi (ed.), Springer, pp. 49-64, 2009 (based on the content of Chapter 6).
4. Tsung-Yuan Liu and Wen-Hsiang Tsai, “Data hiding in graphic drawings by structures of object groupings,” *Journal of Information Science and Engineering*, accepted and to appear (based on the content of Chapter 7).
5. Tsung-Yuan Liu and Wen-Hsiang Tsai, “Quotation authentication: a new approach and efficient solutions by data hiding and cascade hashing techniques,” submitted to *IEEE Transactions on Information Forensics and Security* (based on the content of Chapters 4 and 5).
6. Tsung-Yuan Liu and Wen-Hsiang Tsai, “Active quotation authentication in Microsoft Word documents using block signatures,” *Proceedings of the 3rd International Conference on Information Technology: Research and Education* (ITRE 2005), pp. 260-264, Hsinchu, Taiwan, Republic of China, June 27-30, 2005 (based on the content of Chapter 4).