國立交通大學

資訊科學與工程研究所

碩 士 論 文

利用資訊隱藏技術在 PDF 檔案上做秘密傳輸、驗證及分享之研究

A Study on Information Hiding Techniques for PDF Files and Their Applications for Secret Transmission, Authentication, and Sharing

研 究 生：賴以晟

指導教授：蔡文祥　教授

中 華 民 國　九 十 八 年 六 月

利用資訊隱藏技術在 PDF 檔案上做秘密傳輸、驗證及分享之研究

研究生：賴以晟　　指導教授：蔡文祥 博士

國立交通大學資訊科學與工程研究所

摘要

隨著網路技術的發展，今人皆利用網路來交換資訊。PDF 檔案格式有各種優異特性，如合乎開放標準、跨平台及提供高品質的閱覽與列印，因此已變得非常普及。在本論文中我們利用 PDF 檔案做為掩護檔案，提出數種資訊隱藏的方法，並將其應用在秘密傳輸、驗證以及分享之上。在秘密傳輸方面，我們提出了兩種基於資訊隱藏的傳輸技術。第一種是將秘密資訊編碼，成為 PDF 檔案中代表空白的 16 進位碼，用以取代原本的空白。而第二種則是將文字區塊插入到 PDF 檔案當中，使其不會顯現在 PDF 的瀏覽畫面之中。對於秘密驗證，我們修改 PDF 檔案中的產生數(generation numbers)來藏入驗證訊號，以達到驗證秘密資訊真偽性的目的。在秘密分享方面，我們將秘密 PDF 文件和數個 PDF 掩護檔案，利用“互斥或”(exclusive-OR) 和 “重合”(coincidence)兩種運算，產生一份中間的秘密資訊，並將它藏入這些 PDF 掩護檔案中得到分享檔案。最後我們將這些分享檔案分發給每一位參與者。藉由收集每一份分享檔案，我們可將秘密 PDF 文件復原回來。實驗結果證明以上所提出各方法的可行性。

A Study on Information Hiding Techniques for PDF Files and Their Applications for Secret Transmission, Authentication, and Sharing

Student: Yi-Cheng Lai Advisor: Wen-Hsiang Tsai

Institute of Computer Science and Engineering
National Chiao Tung University

ABSTRACT

With the development of network techniques, people exchange information on the Internet nowadays. The PDF becomes very popular today because of its outstanding capabilities like open-standard, cross-platform, and high-quality viewing and printing. In this study, several data hiding methods for applications of secret transmission, authentication, and sharing are proposed using PDF files as cover media. For secret transmission, we propose a method for secret transmission by two data hiding techniques. The first is a technique based on a space coding scheme and the second is a technique of inserting invisible texts into PDF files. For secret authentication, we propose a method for authentication of secret messages for fidelity by modifying the generation numbers in a PDF file to embed authentication signals of the secret messages. For secret sharing, we use a secret PDF document and cover PDF files to create an intermediate secret message by exclusive-OR and coincidence operations, and hide it into the cover PDF files by the second previously-mentioned data hiding technique to get shares. The shares are then distributed to the participants for them to keep. By collecting all the shares, the secret PDF document can be recovered. Experimental results show the feasibility of the all proposed methods.

ACKNOWLEDGEMENTS

The author is in hearty appreciation of the continuous guidance, discussions, support, and encouragement received from my advisor, Dr. Wen-Hsiang Tsai, not only in the development of this thesis, but also in every aspect of my personal growth.

Thanks are due to Mr. Jiun-Tsung Wang, Mr. Jian-Yuan Wang, Miss. Shu-Hung Hung, Mr. Chun-Pei Chang, Miss Mei-Fen Chen, Miss Chiao-Chun Huang and Miss Chih-Ting Yang for their valuable discussions, suggestions, and encouragement. Appreciation is also given to the colleagues of the Computer Vision Laboratory in the Institute of Computer Science and Engineering at National Chiao Tung University for their suggestions and help during my thesis study.

Finally, the author also extends his profound thanks to his family for their lasting love, care, and encouragement. He dedicates this dissertation to his beloved parents and friends.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction

1.1 Motivation

Information communication plays a very important role in the development of science and technology. Initially, humans exchange information by talking to one another. Later, people write messages on paper and send them to others. Today, people can exchange information on the Internet without face-to-face meeting. The exchange process is time-saving and convenient, but we need to pay attention to the safety and accuracy of the information which we send.

Many kinds of file formats can be used in information communication. The PDF (portable document format) is one of the most common formats and becomes more and more popular nowadays because of its advantages in various applications, such as high-quality printing and cross-platform applicability. In addition to using PDF files directly to exchange information, it is also advantageous to use them as *cover media* to hide data for transmitting secret messages and other uses. Although PDF files are very popular, there are yet not many researches on data hiding in PDF files. It is desirable to have more kinds of data hiding techniques for various application purposes.

Furthermore, it is not safe for people to transmit secret messages on the Internet. The messages which people send may be intercepted and tampered with even if data hiding techniques are used. So it is better to design certain schemes for protections of the security of the transmitted secret message. Furthermore, to check the fidelity and

integrity of the secret messages which are hidden in PDF files, it also is desired to design techniques to authenticate the secret messages.

Sometimes the secret which we want to protect is the PDF file itself. For example, it may happen that three persons want to publish a new research paper and save their paper in a PDF file, like most people do. It is desired to design an information sharing technique to share the PDF file so that before the publication of this paper, only when the three people bring their shares together can the secret PDF file be recovered.

In this study, we want to develop techniques of data hiding, message authentication, and secret sharing for PDF files.

1.2 Overview of Related Works

In this study, some new data hiding techniques in PDF files and their applications are proposed. Before focusing on the methods proposed in this study, we will make an overview of related works in Chapter 2. First, a review of the PDF standard will be described, and then the basic types of objects and file structure of the PDF will be introduced.

Data hiding is a technique to embed concerned data in a cover medium with little or no change on the appearance of the medium so that other people cannot be aware of the existence of the hidden data. Data hiding techniques have been studied for a long time. Some are used in the PDF in recent years.

In addition, secret sharing is a method used to distribute a secret into some shares. In traditional secret sharing, only when a pre-defined threshold number of these shares are collected can the secret be recovered. A more detailed review of techniques for data hiding in PDF files and secret sharing will also be described in Chapter 2.

1.3 Overview of Proposed Methods

1.3.1 Terminology

The definitions of some related terminologies used in this study are described as follows.

1. *Cover media*: cover media, such as images, documents, or videos, are files in which messages are embedded.
2. *White-space characters*: white-space characters are used to separate syntactic constructs from each other in PDF documents.
3. *Cover PDF*: a cover PDF is a PDF document in which messages or signals will be embedded.
4. *Stego-PDF*: a stego-PDF is a PDF document with some secret messages or signals embedded in.
5. *Text matrix*: a text matrix is an object in a PDF document, which is used to describe transformations of the texts in the document.
6. *Generation number*: A generation number of an object in a PDF file is used to keep track of the times for which the object has been updated.

1.3.2 Brief Descriptions of Proposed Methods

1.3.2.1 Secret Transmission via PDF Files by Space Coding and Insertion of Invisible Texts

In this study, we propose two data hiding techniques via PDF files for secret transmission. The first is based on a *space coding* scheme. When opening a PDF file with some text editor, we can see the initial code of the PDF file, in which there are

many *white-space characters* used to separate syntactic constructs from one another. According to the PDF standard, many distinct characters are treated as *white-space characters*. We can use this property to embed secret messages. First, we encrypt a given secret message with a user key and then embed it into a cover PDF by space coding so that we can get a stego-PDF. A sender can then send this stego-PDF to a receiver. When the receiver wants to get the secret message, he/she should extract it from the stego-PDF with a correct user key. An advantage of the proposed method is that it will not change the size of the cover PDF.

The second method is based on a scheme of inserting invisible texts into a cover PDF. In the PDF, a *text matrix* is used to describe transformations of the texts. We can insert *text matrices* with their coordinates being outside the visible area of the PDF so that the corresponding text will not show on the displayed PDF documents. The basic idea is to encode each secret message as decimal numbers, put them in text matrices, and process these numbers to make sure that the position of texts is outside the visible area. Since the embedded data will not appear in the displayed document, the capacity of the data which we embed is *unlimited*.

Furthermore, we can combine the two methods together to enhance data hiding effects. More details will be described in Chapter 3.

1.3.2.2 Authentication of Secret Messages for Fidelity Verification in PDF Files

In order to check if a secret message has been tampered with or not, we propose a method to authenticate the secret message for fidelity verification. Since what we want to authenticate is a secret message, we need to embed authentication signals by a different way from that we use to embed the secret message. In the PDF, a *generation*

number of an object is used to keep track of the times for which the object has been updated. It has no influence on the display of the PDF file. So, we propose to alter *generation numbers* to embed authentication signals.

First, we use a secret message to generate authentication signals and then embed them into the *generation numbers* of some objects of the PDF files. When extracting the secret message from a stego-PDF, we generate the authentication signals again and compare them to the signals which we embedded in the stego-PDF. If they are *not* the same, it indicates that the secret messages might have been tampered with by some illicit user and the secret message is *not* believable. More details will be described in Chapter 4.

1.3.2.3 Secret Sharing via PDF Documents by Data Hiding Techniques

In this study, we propose a method for secret sharing via PDF documents. Given a secret PDF document, first we prepare *n* PDF documents randomly. From each PDF document, we extract a portion of data with the same size as the secret PDF file. Every portion of data is regarded as a *preliminary share*. Then we use these preliminary shares and the secret PDF document to generate an *intermediate secret message* by using exclusive-OR and coincidence operators randomly. And then we use the data hiding techniques proposed in this study to hide the intermediate secret message in the *n* PDF documents. In this way, we get *n* stego-PDFs as shares for each participant to keep one of them.

When we want to recover the secret PDF document, we need *every* stego-PDF from the participants. We extract the intermediate secret message and preliminary shares from these stego-PDF files, and then use them to recover the secret PDF

document. More details will be described in Chapter 5.

1.4 Contributions

Several contributions are made in this study, as described in the following:

1. A data hiding method for PDF documents with the advantage of preserving the size of the PDF document is proposed.
2. Another data hiding method for PDF documents with unlimited data hiding capacity is proposed.
3. A secret transmission method via PDF documents based on the use of the proposed data hiding techniques is proposed.
4. A secret authentication method for PDF documents is proposed for verifying the fidelity of secret messages.
5. A secret sharing method based on the use of the proposed data hiding techniques for PDF documents is proposed.

1.5 Thesis Organization

In the remainder of this thesis, a review of related works about data hiding, secret sharing, and the PDF standard is described in Chapter 2. In Chapter 3, the proposed method for secret transmission is described. In Chapter 4, the proposed secret authentication method for fidelity verification in PDF documents is described. In Chapter 5, the proposed secret sharing method for PDF documents based on the proposed data hiding techniques is described. Finally, conclusions and some suggestions for future works are given in Chapter 6.

Chapter 2
Review of PDF Standard and Survey of Related Works

2.1 Introduction

Since we use PDF files as cover media to implement the data hiding techniques which we proposed in this study, we need to know the framework of the PDF first. A review of the PDF standard will be described in Section 2.2.

In Section 2.3, a review of existing techniques for data hiding in PDF files will be described, and then a review of existing techniques for secret sharing is presented in Section 2.4.

2.2 Review of PDF Standard

2.2.1 Overview

The Adobe portable document format (PDF) is a file format of the Adobe® Acrobat® family of products [1]. The contents in PDF files are described by a context-free grammar which is modified from PostScript®. PDF is a file format for representing documents in a manner independent of the application software, hardware, and operating systems used to create them and of the output device on which they are to be displayed or printed.

The basic elements in the PDF are objects. A PDF document consists of a

collection of objects that together describe the appearance of one or more pages of the PDF. A document's pages can contain any combination of text, graphics, and images. A page's appearance is described by a PDF content stream, which contains a sequence of graphics objects to be painted on the page.

2.2.2 Basic Types of Objects

A PDF document is a data structure composed from a small set of basic types of data objects. Objects may be labeled so that they can be referred to by other objects. A labeled object is called an indirect object. PDF supports eight basic types of objects: Boolean values, integer and real numbers, strings, names, arrays, dictionaries, streams, and the null object. Each object type and the indirect object are briefly described below.

1. **Boolean Objects**

   The PDF provides boolean objects identified by the keywords *true* and *false*.

2. **Numeric Objects**

   The PDF provides two types of numeric objects: integer and real. The range and precision of numbers are limited by the internal representations used in the computer on which the PDF consumer application is running. For example, 17, +99, −3. 18.9, −.005, 0.0, and 5. are all numeric objects.

3. **String Objects**

   A string object consists of a series of bytes — unsigned integer values in the range 0 to 255. The length of a string may be subject to implementation limits. String objects can be written in two ways:

(1) As a sequence of literal characters enclosed in parentheses ( ). For example: (This is a string).

(2) As hexadecimal data enclosed in angle brackets < >. For example: < 762073686D602E >.

4. Name Objects

A name object is a symbol uniquely defined by a sequence of characters. A slash character (/) introduces a name. The slash is not part of the name but is a prefix indicating that the following sequence of characters constitutes a name. For example: /NameOne.

5. Array Objects

An array object is a one-dimensional collection of objects arranged sequentially. An array's elements may be any combination of numbers, strings, dictionaries, or any other objects, including other arrays. An array is written as a sequence of objects enclosed in square brackets ([and]). For example: [true 17.417 (Hello World) /SomeName].

6. Dictionary Objects

A dictionary object is an associative table containing pairs of objects, known as the dictionary's entries. The first element of each entry is the key and the second element is the value. The key must be a name and the value can be any kind of object, including another dictionary. A dictionary is written as a sequence of key-value pairs enclosed in double angle brackets (<< ⋯ >>). For example: << /Type /Example

/Version 3.02

/String1 (Hello World)

/Array1 [1 0 0 1]

>>

7. Stream Objects

A stream object, like a string object, is a sequence of bytes. But it's unlike a string is subject to an implementation limit; a stream can be of unlimited length. A stream consists of a dictionary followed by zero or more bytes bracketed between the keywords *stream* and *endstream*. For examples:

<<…>>

stream

…Zero or more bytes…

endstream

8. Null Object

A null object has a type and value that are unequal to those of any other object. There is only one object of type null, denoted by the keyword *null*.

2.2.3 Indirect Objects

Any object in a PDF file may be labeled as an indirect object. This gives the object a unique object identifier by which other objects can refer to it. The object identifier consists of two parts as follows.

1. A positive integer *object number*. Indirect objects are often numbered sequentially within a PDF file, but this is not required; object numbers may be assigned in any arbitrary order.
2. A non-negative integer *generation number*. In a newly created file, all indirect objects have generation numbers of 0. Nonzero generation numbers

may be introduced when the file is later updated.

The combination of an object number and a generation number uniquely identifies an indirect object. The definition of an indirect object in a PDF file consists of its object number and generation number, followed by the value of the object bracketed between the keywords *obj* and *endobj*. The object can be referred to from elsewhere in the file by an *indirect reference* consisting of the object number, the generation number, and the keyword *R*. An example of indirect objects is as follows:

5 0 *obj*

<< /Length 3 0 *R* >>

stream

…

endstream

endobj

The example defines an indirect object with an object number of 5, a generation number of 0 and a stream object with its dictionary. According to the dictionary of the stream object, the length of the stream refers to another indirect object which has an object number of 3 and a generation number of 0.

2.2.4 File Structure

In this section, we will describe how objects are organized in a PDF file. A canonical PDF file initially consists of four elements (see Figure 2.1) briefly described as follows.

1. A one-line *header* identifying the version of the PDF specification to which the file conforms.

   For example, for a file conforming to PDF 1.4, the header should be

"%PDF-1.4."

2. A *body* consists of a sequence of indirect objects representing the contents of a document.
3. A *cross-reference table* containing information about the indirect objects in the file.

The cross-reference table is the only part of a PDF file with a fixed format. The table comprises one or more cross-reference sections. Each section begins with a line containing the key word xref. Following this line are one or more cross-reference subsections. Each subsection begins with a line containing two numbers separated by a space: the object number of the first object in this subsection and the number of entries in the subsection. Following this line are the cross-reference entries themselves, one per line. Each entry containing a 10-digit byte offset, a 5-digit generation number and a key word n (for an in-use entry) or f (for a free entry). The 10-digit byte offset gives the number of bytes from the beginning of the file to the beginning of the object when the entry is in use. If the entry is free, it gives the object number of the next free object. The following example has 7 cross-reference entries, object 12 is the first object and its offset is 16. Object 15 and 17 are free.

xref
12 7
0000000016 00000 n
0000000321 00000 n
0000000972 00000 n
0000000017 00007 f
0000001314 00000 n
0000000000 00003 f
0000001843 00001 n

4. A *trailer* giving the location of the cross-reference table and of certain special objects within the body of the file.

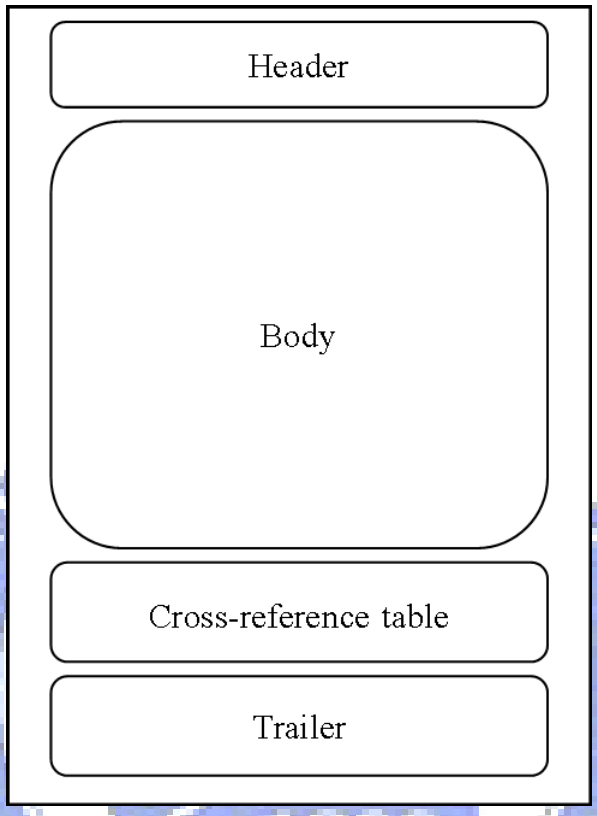


Figure 2.1 Initial structure of a PDF file.

2.3 Review of Techniques for Data Hiding in PDF files

Techniques for data hiding have been studied for a long time and applied to various media of various image formats, videos, and documents [2-9]. The PDF becomes one of the most popular formats for people to exchange information nowadays and several techniques for data hiding in PDF files have been proposed in recent years [10-13].

Zhong and Chen [10] proposed an information steganography algorithm on PDF

documents by hiding data between indirect objects of PDF documents. The algorithm can embed data of unlimited lengths into PDF documents and the embedded PDF documents keep transparent when being displayed in PDF readers. Zhong, Cheng and Chen [11] proposed a steganographic technique for hiding data in a kind of PDF English texts. They modified integer numerals which are used to position characters in the PDF text. Because the perceptual difference is very small, people cannot be aware of the hidden data in the PDF document. Liu et al. [12] proposed an algorithm based on equivalent transformations in PDF files. They discovered that the effect of the page display of a PDF file is extraneous to the seriation of the dictionary's entries so that data hiding can be achieved by special array of entries, instead of by operations of adding any other data to the cover PDF. Wang and Tsai [13] proposed a data hiding method by slight modifications of the values of various PDF object parameters, yielding a difference of appearance very difficult to notice by human vision.

2.4 Review of Techniques for Secret Sharing

Secret sharing is a method for distributing a secret into several shares which are then distributed to some participants. Each of them keeps one of the shares and every share is meaningless alone. Only when a pre-defined threshold number of these shares are collected together can the secret be recovered.

Blakley [14] was the first to publish an approach to solving the secret sharing problem. His is a probabilistic approach based on linear projective geometry. Each v_i specifies a hyperplane and the secret s is the unique point of intersection of the n hyperplanes. Shamir [15] proposed a simple and efficient secret sharing scheme

which is called a (k, n)-threshold scheme, where k is the threshold number of how many shares should be collected at least and n indicates the number of participants. Lin and Tsai [16] proposed an efficient (n, n)-threshold method by using exclusive-OR operations. It simply applies the exclusive-OR operation to a secret image as well as $n - 1$ other images to generate the nth image. The $n - 1$ images and the nth image are all regarded as shares and are distributed to n participants, respectively. The secret image can be recovered only by exclusive-ORing the n images which are kept by the n participants. Huang and Tsai [17] [18] proposed secret sharing methods for pure texts, HTML documents, and e-mail documents. For pure texts, they transformed a secret text into several shares, which are meaningful articles and can be authenticated. For HTML documents, the method was designed to extract important parts of the components in a secret HTML document and share them, and then transform the share data into an HTML document of the same appearance of the secret HTML document so that each share is still a HTML-type share. For e-mail documents, a secret e-mail is encoded and distributed into several authenticable e-mail shares by hierarchical sharing with data magnitude control, steganography methods, and authentication techniques. In addition, a secret H.264/AVC video sharing scheme was proposed by Huang and Tsai [19] to extract prediction modes from given cover videos and the secret video, then share the intra-prediction modes of the secret video based on the exclusive-OR operation, and hide finally the resulting share data into the prediction modes of the cover videos.

Chapter 3
Secret Transmission via PDF Files by Space Coding and Insertion of Invisible Texts

3.1 Introduction

With the development of network techniques, people can exchange information by writing e-mails or sending files on the Internet. Secret messages can even be transmitted by data hiding techniques, using some types of files as cover media. There are many data hiding methods for secret transmission via images, videos, and other types of files. Because the PDF has become a very popular file format nowadays, it also is a suitable type of file for use in secret transmission as cover media.

The proposed method for secret transmission via PDF files is described in this chapter. In Section 3.2, the proposed two data hiding techniques are described. In Section 3.3, the proposed data hiding processes are described, and the recovery processes are stated in Section 3.4. In Section 3.5, several experimental results are shown and a summary and some discussions of the proposed method are made in Section 3.6.

3.2 Data Hiding Techniques in PDF Files

In this section, the proposed two different kinds of data hiding techniques are

described. The first is based on a space coding scheme and the other is based on a scheme of inserting invisible texts into a cover PDF.

3.2.1 Data Hiding in PDF Files by Space Coding

As mentioned in Chapter 1, white-space characters are used to separate syntactic constructs from one another in the PDF file. According to the PDF standard, many distinct characters are treated as white-space characters. Table 3.1 shows this property of the PDF.

Table 3.1 White-space characters.

Decimal	Hexadecimal	Name
0	00	Null (NUL)
9	09	Tab (HT)
10	0A	Line feed (LF)
12	0C	Form feed (FF)
13	0D	Carriage return (CR)
32	20	Space (SP)

All white-space characters are equivalent, except in comments, strings, and streams. So that if white-space characters are not in comments, strings, and streams, they all available as white spaces in PDF documents. We can use this property to embed secret messages into cover PDFs.

More specifically, according to Table 3.1 a white-space character may have six

kinds of codes, which are the hexadecimal numbers 00, 09, 0A, 0C, 0D, and 20. After some experiments, we found out that in some text editors, 0C will show as a line and 0A will cause line feeding on the display of the original code of a PDF document. If we use these two codes to embed secret messages, people may be aware of the existence of the hidden data more easily. So we only use 00, 09, 0D and 20 to embed the secret message. Thus, if the four kinds of white-space characters are not in comments, strings, and streams, they are all *usable* to embed the secret message.

Accordingly, a white-space character has four different codes to use now, so we can embed 2 bits of message data using a single white-space character. More specifically, we use the hexadecimal codes 00, 09, 0D and 20 to represent the 2-bit message data 00, 01, 10, and 11, respectively.

Table 3.2 Proposed data encoding scheme.

Data to embed	Hexadecimal code
00	00
01	09
10	0D
11	20

For example, if the secret message is 0011011110, we need to use five white space characters to embed the message and the codes of them are 00, 20, 09, 20, and 0D, respectively.

Besides, in order to tell how many white-space characters have been modified for data hiding, we use two bytes to embed the length of the secret message before embedding the secret data. These two bytes will be used in the message data recovery process.

Since we do not insert any other data into the cover PDF in the above-described message data embedding process, the size of the PDF file will not change so that it is difficult for a reader of the displayed stego-PDF file to be aware of the existence of the hidden data. But obviously the capacity of the data which we embed, called *data embedding capacity* in the sequel, is limited by how many usable white-space characters does the cover PDF have.

3.2.2 Data Hiding in PDF Files by Insertion of Invisible Texts

In this section, the other proposed data hiding technique based on a scheme of inserting invisible texts into a cover PDF is described. The basic idea is that we can insert into the PDF file some text matrices whose coordinates are outside the visible area of the PDF so that the corresponding texts will not show on the displayed PDF document. The details are described as follows.

By definition, a text matrix is used to set the state of the corresponding text and locate it in a PDF file. The structure of a text matrix is shown below:

a b c d e f Tm

where *a* through *f* are all numbers and Tm indicates the end of the text matrix. The first four numbers "*a b c d*" are used for text scaling, rotation, and skew in the following way.

1. Scaling is obtained by "*Sx* 0 0 *Sy*". This scales the coordinates so that 1 unit in the horizontal and vertical dimensions of the new coordinate system is the same size as *Sx* and *Sy* units, respectively, in the previous coordinate system.

2. Rotations are produced by "$\cos\theta$ $\sin\theta$ $-\sin\theta$ $\cos\theta$", which has the effect of rotating the coordinate system axes by an angle θ counterclockwisely.
3. Skew is specified by "1 $\tan\alpha$ $\tan\beta$ 1", which skews the *x*-axis by an angle α and the *y*-axis by an angle β.

The initial values of *a*, *b*, *c* and *d* are 1, 0, 0 and 1, respectively. The other two numbers *e* and *f* are the distances to translate the origin of the coordinate system in the horizontal and vertical dimensions, respectively. The proposed method is designed to transform secret data into decimal numbers. The transformation specified by Table 3.3. After the transformation, the secret data become a long string of decimal digits, which we regard as a big integer number. Then we process this integer number, say denoted as *N*, by distinct ways according to the following three conditions.

1. Set *e* to *N* if *N* does not cause an overflow and set *f* to 0. (Note that, as mentioned in Section 2.2.2, the range and precision of numbers are limited by the internal representations used in the computer on which the PDF consumer application is running. So when we set *e* to *N*, an overflow may occur.)
2. If *N* causes an overflow, separate *N* into two numbers $N_1$and N_2. Let *N* be $n_1n_2n_3\ldots n_kn_{k+1}\ldots n_l$, where n_i is a decimal digit, *i* is from 1 to *l*, *l* is the number of decimal digits in *N*, and *k* is the largest number which does not make *N* causing an overflow. Then, set $n_1n_2n_3\ldots n_k$ as N_1 and $n_{k+1}\ldots n_l$ as N_2; and treat N_1 as *e* and N_2 as *f*.
3. If either N_1 or N_2 or both of them cause overflows, insert more text matrices to embed the secret data.

Table 3.3 The transformation between binary bits and decimal numbers.

Bit stream	Decimal number	Bit stream	Decimal number
000	1	100	5
001	2	101	6
010	3	110	7
011	4	111	8

In a PDF document, each page of the document is represented by a *page object*—a dictionary that includes references to the page's contents and other attributes. Each page object has a parameter named *MediaBox* which defines the boundaries of the physical medium on which the page is intended to be displayed or printed. In short, the MediaBox decides the visible area of the page. A common visible area of a PDF page is 595×842. In order to guarantee that the position of texts is outside the visible area to create invisibility to the observer, we concatenate "999" before the decimal numbers. An example is given as follows.

Suppose the secret message is "010001110." Then, according to Table 3.3, we can transform it into "327". And then we concatenate "999" before "327", yielding "999327". Finally, we insert a text matrix and put "999327" in it. The final text matrix is shown below:

1 0 0 1 999327 0 Tm.

In addition, in order to reduce the file size, the PDF supports two compression filters, LZW and Flate compressions, for the content streams describing texts and graphics in the PDF document. So when we want to insert the text matrices, we need to decompress the page's content stream first, and after inserting the text matrices in

which we embed the secret message, we should compress the modified page's content stream again by the default compression filter.

Because we insert more data in the cover PDF, the offset of each indirect object and the offset of the cross-reference table may change. A wrong offset of the indirect objects in the cross-reference table and trailer can cause a wrong display of the PDF document. So we need to update the cross-reference table and trailer of the cover PDF to get the desired stego-PDF. More specifically, suppose we embed the secret message in an indirect object B to get B'. Since we insert more data in B, the size of B' is bigger than B. So the offsets of the indirect objects whose location are behind B need to be updated by increasing them for a value D which is the difference of the size between B and B'. And if the cross-reference table is also behind B, the trailer needs to be updated by the same way, too.

Since the embedded data will not appear in the displayed PDF document, the capacity of the data which we embed is unlimited, at the sacrifice of the resulting size of the stego-PDF file.

3.3 Proposed Data Hiding Algorithm

We have proposed two data hiding techniques via PDF files. We may use only one of them or both for secret transmission. For the latter case, if the secret message is short enough, then the first technique suffices to embed all the message data into the cover PDF; or if we cannot embed data just by the first technique, then the second technique is used further to embed the remaining data. The flowchart of this *combined* data hiding method is illustrated by Figure 3.1 and the detail is described below as an algorithm.

Algorithm 3.1. Encoding a message and hiding the result in a PDF file.

Input: a user key K, a secret message S, and a cover PDF P.

Output: a stego-PDF P'.

Steps:

1. Encrypt S by some method with K, for example by the DES algorithm, to get encrypted secret data S' and let l be the length of S' in bytes.
2. Find out the number m of all usable white-space characters to get the capacity n in bytes as $n = m/4$.
3. Separate S' into two parts S_1 and S_2 by the following way.
   3.1 If $n \geq l+2$, take S_1 to be S' concatenated by its length information and S_2 to be null.
   3.2 Else, take S_1 to be the first $n - 2$ bytes of S' concatenated by its length information and S_2 to be the remaining bytes of S'.
4. Embed S_2 into P by the following way.
   4.1 If S_2 is not null, transform it into 3-bit groups (if the number of bits is not divisible by 3, pad 1 or 2 zeros after the last group).
   4.2 Map the 3-bit groups into decimal numbers $g_1, g_2 \ldots, g_k$ by Table 3.3.
   4.3 Embed g_1 through g_k into P by the way of inserting them as invisible texts, as mentioned before in Section 3.2.2.
5. Embed S_1 into P by the following way.
   5.1 Divide each 2 bits of S_1 into *groups of bitstream* $f_1, f_2 \ldots, f_{4w}$, where w denotes the length of S_1.
   5.2 Transform f_1 through f_{4w} into hexadecimal codes $h_1, h_2 \ldots, h_{4w}$, respectively, by Table 3.2.
   5.3 Replace the code of each usable white-space character by h_1 through h_{4w}.
6. Update the cross-reference table and trailer, if needed, to get P'.

3.4 Proposed Data Recovery Algorithm

We have introduced the proposed data hiding algorithm in the last section. After implementing the algorithm, we can get a stego-PDF with a secret message embedded in it. A sender can then send this stego-PDF to a receiver. When the receiver wants to recover the secret message, he/she should extract it from the stego-PDF with a correct user key. For this, we should find out all usable white-space characters first. Next, we extract the length of the hidden data by decoding the first eight usable white-space characters. And then we decode the hidden data in the white-space characters and concatenate the data, which we embed in the PDF file by the way of inserting invisible texts, to recover the whole encrypted data. Finally, we decrypt the data by the user key and get the secret message. A flowchart of the proposed data recovery algorithm is shown in Figure 3.2 and the detail is described below.

Algorithm 3.2. Recovering a message from a stego-PDF.

Input: a user key K and a stego-PDF P'.

Output: a secret message S.

Steps:

1. Find out all usable white-space characters in P' and decode the first eight characters by Table 3.2 to get the number n of the characters embedded in P'.
2. Get the hexadecimal codes h_1, h_2…, h_n of the n white-space characters which are used to embed S.
3. Transform h_1 through h_n into secret data S_1 by Table 3.2.
4. Scan P' and find out all text matrices inserted and extract the decimal numbers g_1, g_2…, g_k from them.
5. Transform g_1 through g_k into secret data S_2 by Table 3.3.

6. Concatenate S_1 and S_2 into S'.

7. Decrypt S' with K to recover the desired secret message S.

3.5 Experimental Results

In order to implement the proposed method for secret transmission, we designed a user interface for the program we have written in the language of Java. It supports the following three ways to hide and recover secret messages.

1. Using the proposed space coding technique only.
2. Using the proposed technique of inserting invisible texts only.
3. Using the proposed method of combining the two techniques together.

By the way, we can see the function of authentication in the user interface. It is used for authenticating the secret message which we embedded in the PDF file. Authentication of secret messages will be described investigated in the next chapter, so we will not discuss it in this chapter.

Figure 3.3 shows the window of the user interface with a secret message and a user key as input, which means the user embeds the secret message in a cover PDF by the proposed combined method. The result of the secret recovery with a correct user key is shown in Figure 3.4. The cover PDF is shown in Figure 3.5 and the stego-PDF is shown in Figure 3.6. Comparing the two figures, no change can be seen on the display of the PDF document. In addition, Figure 3.7 shows the result of the secret recovery with an incorrect user key.

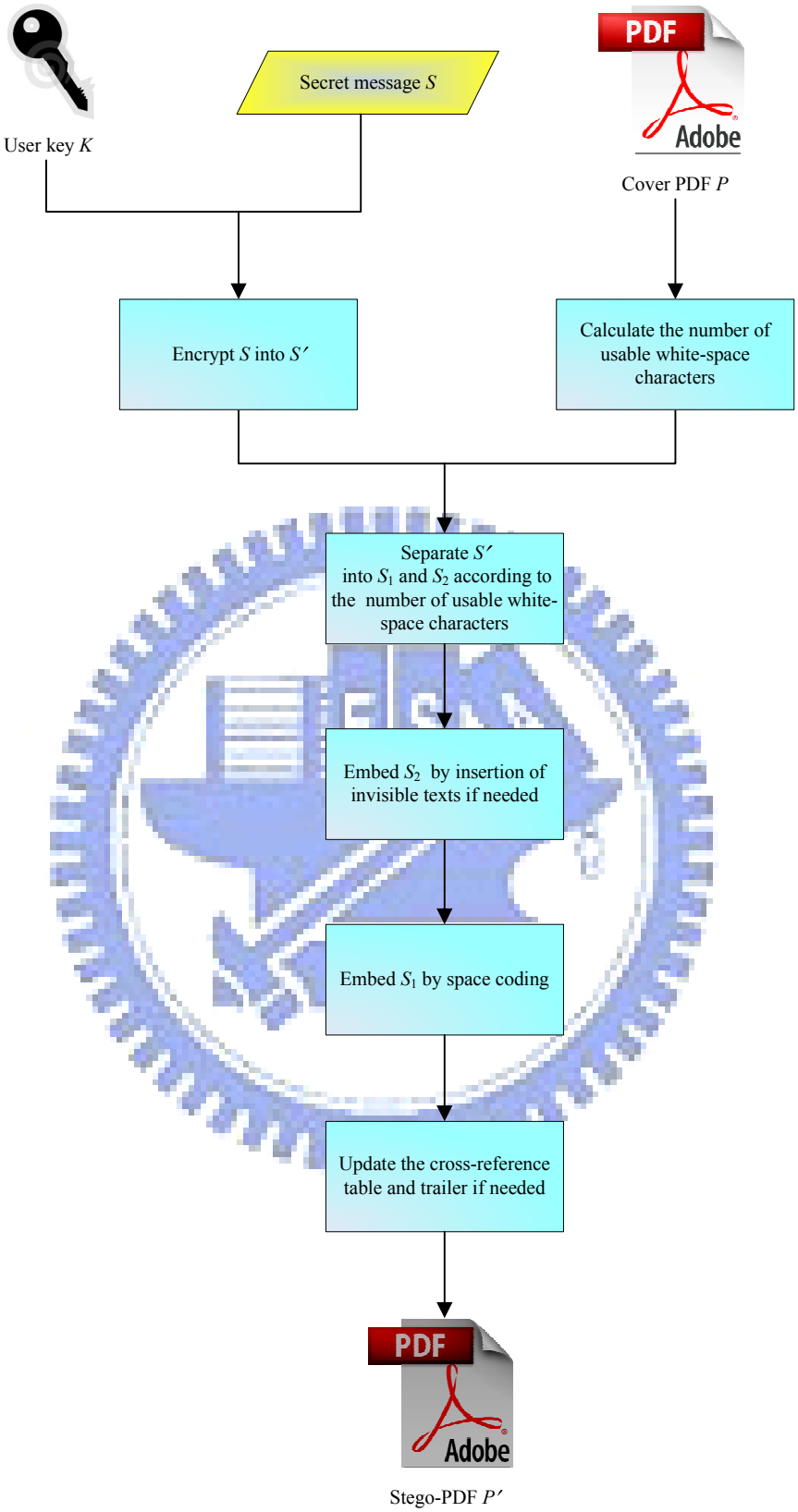


Figure 3.1 The flowchart of the proposed data hiding algorithm.

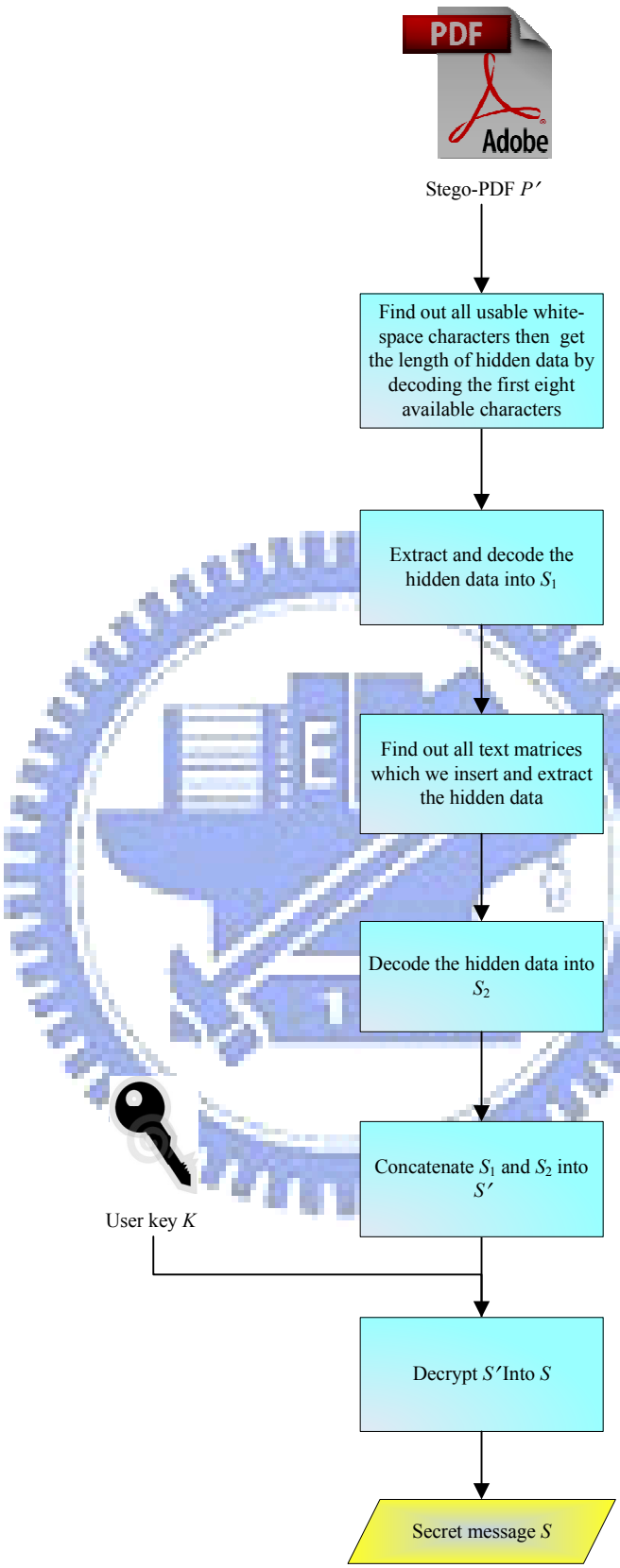


Figure 3.2 The flowchart of the proposed data recovery algorithm.

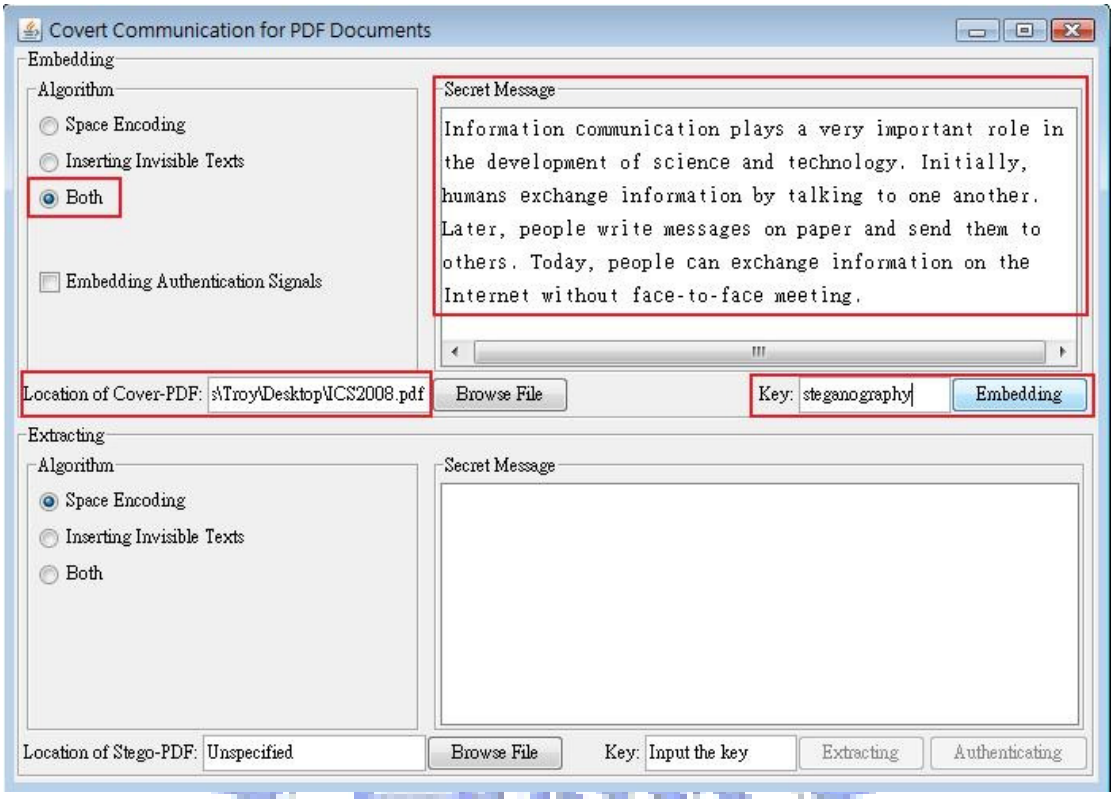


Figure 3.3 The window of the user interface.

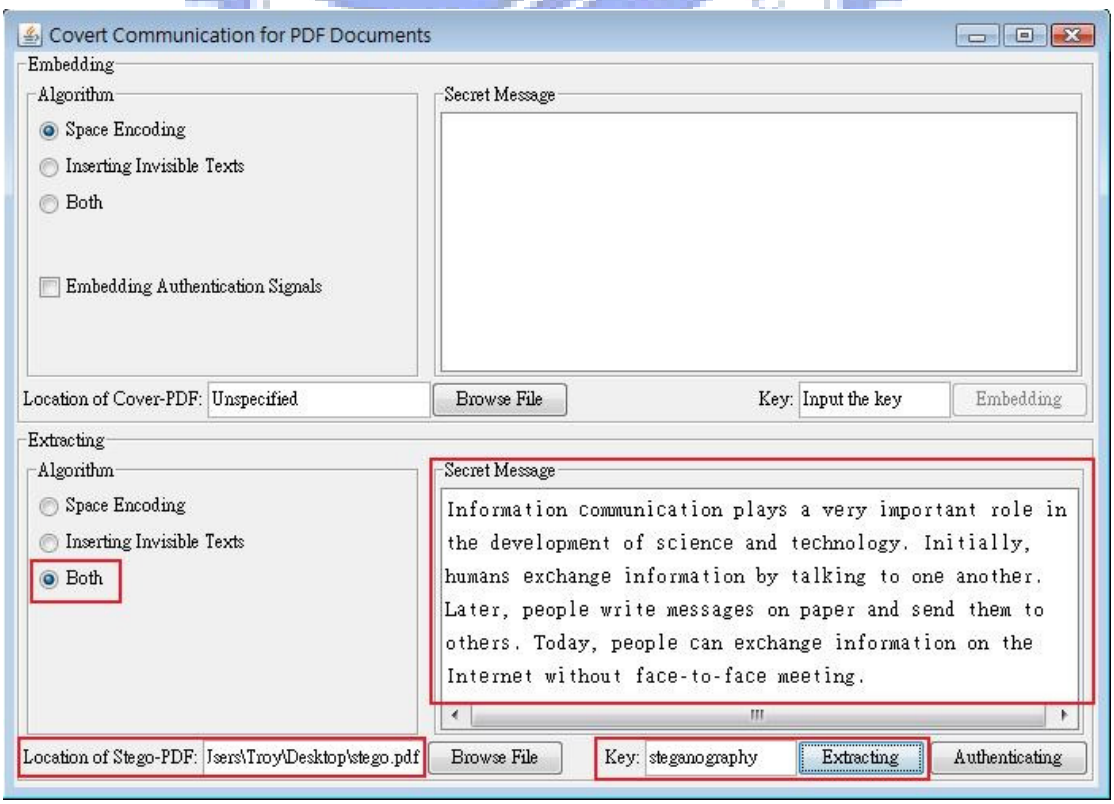


Figure 3.4 The result of secret recovery with the correct user key.

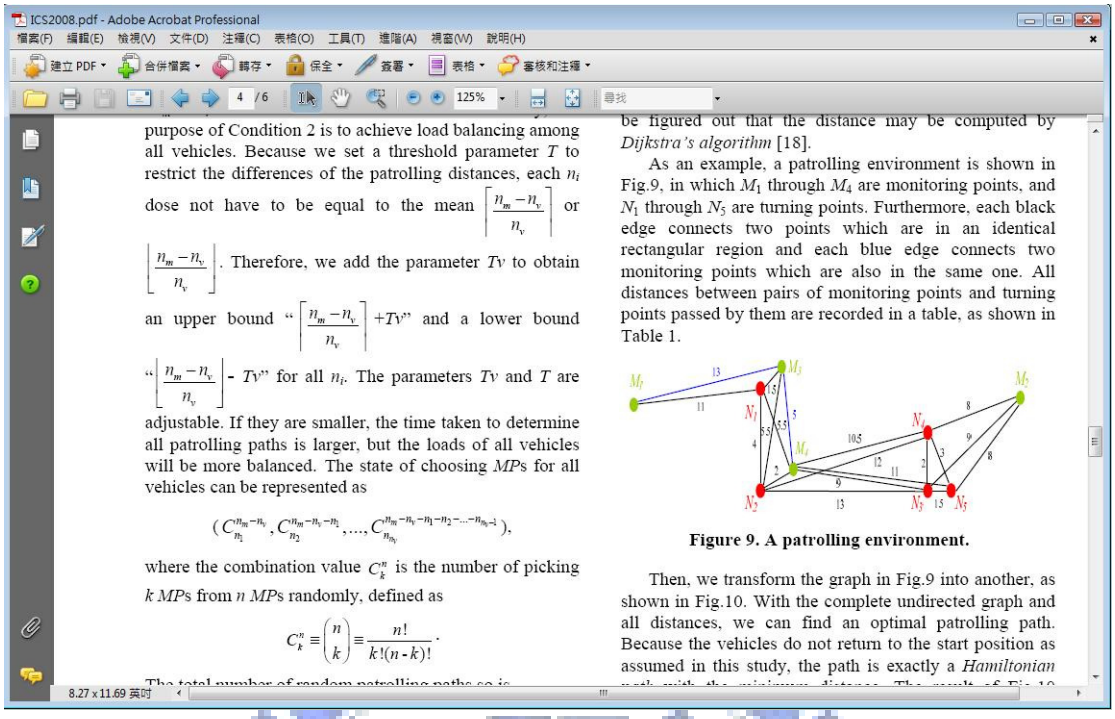


Figure 3.5 The cover PDF.

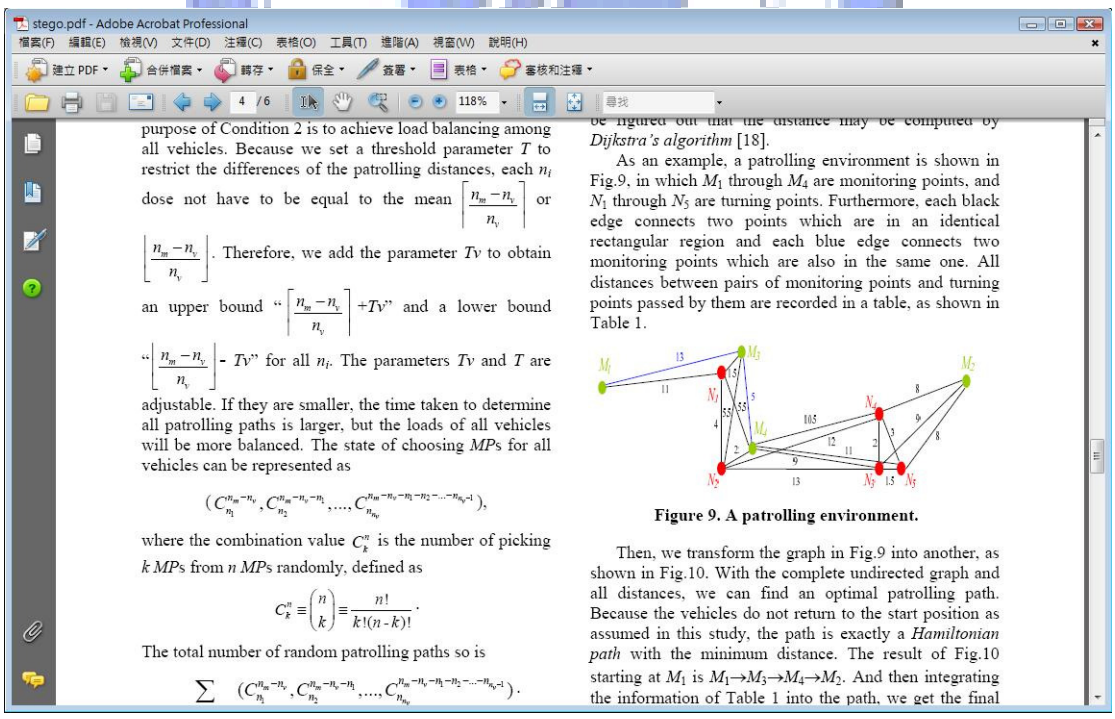


Figure 3.6 The stego-PDF.

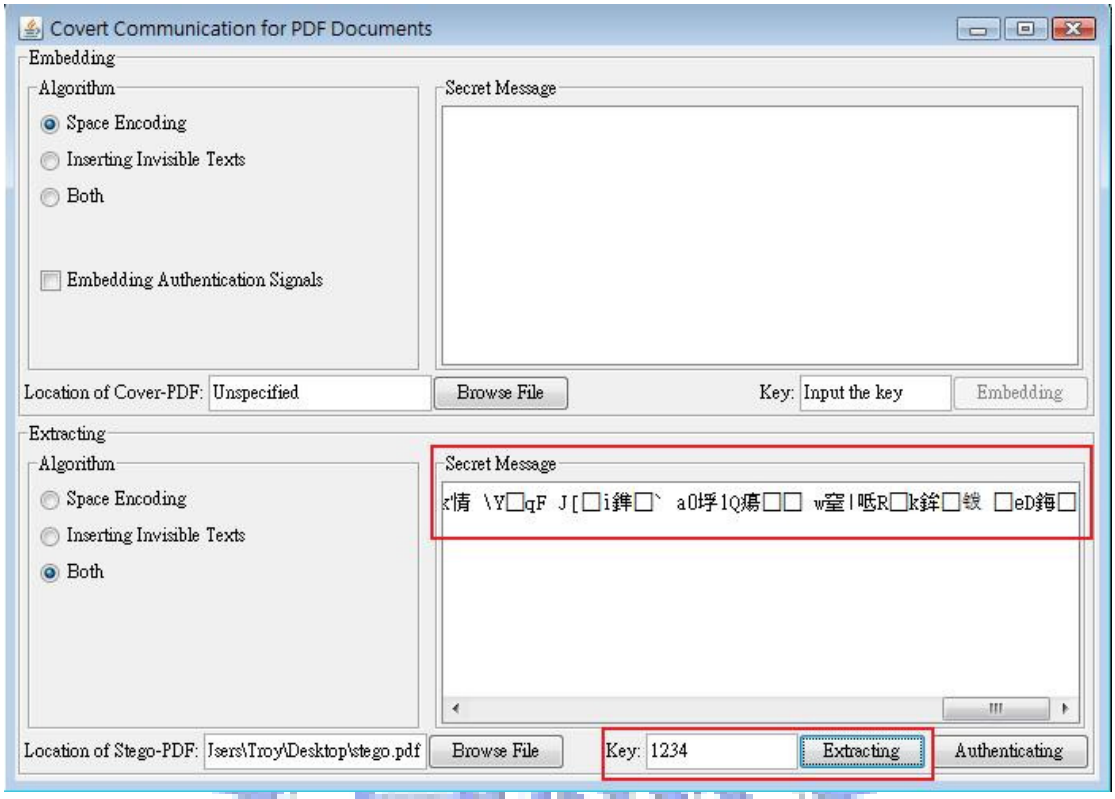


Figure 3.7 The result of secret recovery with the incorrect user key.

3.6 Discussions and Summary

In this chapter, we have proposed two data hiding techniques via PDF files for secret transmission. The first is based on a space coding scheme and the second is based on a scheme of inserting invisible texts into a PDF file.

For the first technique, the capacity of the embedded data is limited by the number of usable white-space characters in the cover PDF while the size of the cover PDF will not change after data embedding because we do not insert any other data in it. For the second technique, the capacity of the embedded data is unlimited so that we can embed a large amount of secret data, which even can be another PDF file. That is just the property we utilize to implement secret PDF document sharing which will be described in Chapter 5.

Furthermore, we can combine these two techniques together for secret

transmission. If the secret message is short enough, we just use the first technique, which does not change the size of the cover PDF; else, the second technique is used subsequently to embed the remaining secret data.

No matter which technique we use to implement secret transmission, or both of them are used, it has no influence on the display of the PDF file so that people cannot be aware of the existence of the hidden data. Even if an illicit user knows that there is a secret message in the PDF file, the covert message can be protected by a user key, and the illicit user still cannot extract the original secret message. The idea has been proved by our experiments.

Chapter 4
Authentication of Secret Messages for Fidelity Verification in PDF Files

4.1 Introduction

It is not safe for people to transmit messages on the Internet because the messages might be intercepted and tampered with by illicit users. Even if data hiding techniques may be used to embed a secret message, the cover medium in which the secret message is embedded still has chance to be altered. So we designed a method in this study to authenticate the secret message to check if it is believable.

More specifically, the proposed method can be used to authenticate a secret message embedded in a PDF file for fidelity verification. The basic idea will be described in Section 4.2. Basically, we modify the generation numbers in the PDF structure as authentication signals. In Section 4.3, the proposed authentication algorithm for secret messages in PDF files will be stated. Several experimental results will be shown in Section 4.4, and then a summary of the proposed method and some discussions are given in Section 4.5.

4.2 Idea of Secret Message Authentication by Modifying

Generation Numbers in the PDF File

Since what we want to authenticate is a secret message, we need to embed authentication signals by a different way from that we use to embed the secret message. The proposed data hiding technique for embedding authentication signals is to modify the generation numbers in the PDF file.

As mentioned in Section 2.2, an identifier of an indirect object in a PDF file consists of two parts: a positive integer object number and a non-negative integer generation number. The generation number is used to keep track of the times for which the object has been updated. It also appears in the cross-reference table.

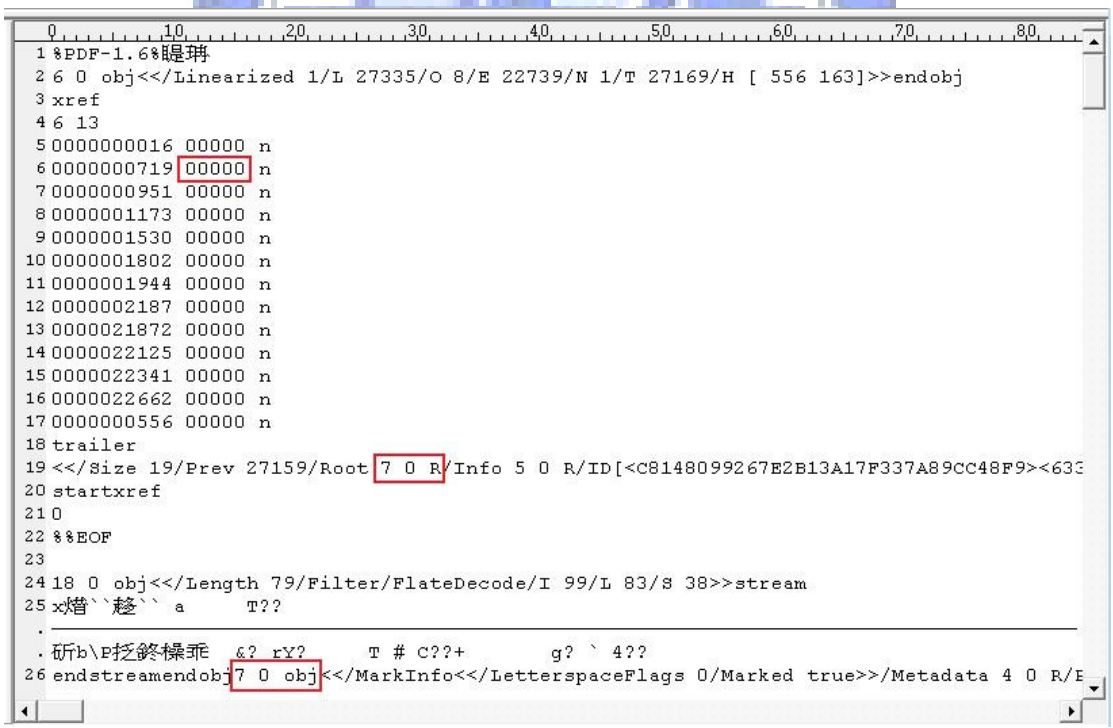


Figure 4.1 The example of generation numbers.

More specifically, in Figure 4.1 we can see a cross-reference table. The second object in the table is object 7, its offset is 719, and its generation number is 0 on

which we use a red rectangle to highlight. Except the cross-reference table, we can know the generation number of object 7 in other places like “7 0 R”, which is used to refer to the object 7, and “7 0 obj”, the identifier of object 7. All generation numbers of the same indirect object in the PDF file must be the same value or it can cause a wrong display of the PDF document.

After some experiments, we find out that the value of a generation number of an indirect object has nothing to do with the display of the PDF document. So we can modify some generation numbers to embed authentication signals. First, we use the secret message to generate authentication signals by applying exclusive-OR operations on each byte of the secret message. Then, we transform the authentication signal into a bitstream and then use each bit in the bitstream to replace the generation numbers of some indirect objects in the PDF file.

For example, suppose the authentication signal is “10011000.” Then, we use each bit of the authentication signal to replace the generation numbers of object 1 to 8, respectively. So the modified generation numbers of these indirect objects are 1, 0, 0, 1, 1, 0, 0 and 0, respectively. Note that all the generation numbers of the same indirect object in the PDF file must be the same value, so we need to modify all generation numbers of the indirect objects which we use to embed the authentication signal.

When we get a stego-PDF, extract a secret message from it, and want to know if the secret message is believable, we use the secret message which we extract from the stego-PDF to generate the authentication signal by the same way. Then we compare it with the signal which is extracted from the generation numbers in the PDF file. If they are the same, the authentication is successful. Else, the secret message is not believable and it is decided to have been tampered with by illicit users.

4.3 Proposed Authentication Processes for Secret Messages in PDF files

In the proposed authentication process, we use the generation numbers of objects 1 through 16 to embed every two bytes of the authentication signal. We separate the authentication process into two parts, which are described here. The first part is the embedding of the authentication signal. Its flowchart is illustrated by Figure 4.2 and the detail is described below as an algorithm.

Algorithm 4.1. Generating and embedding authentication signals in PDF files.

Input: a secret message S and a cover PDF P.

Output: a stego-PDF P'' with an authentication signal for the secret message embedded.

Steps:

1 Apply exclusive-OR operations on all bytes of S to get an authentication signal A by the following way.

1.1 Transform S into $S = s_1s_2s_3\ldots s_n$, where s_1 through s_n are the bytes of S and n is the length of S in bytes.

1.2 Compute

$A_1 = s_2 \oplus s_4 \oplus s_6 \ldots \oplus s_{2k}$, where $k = n/2$;

$A_2 = s_1 \oplus s_3 \oplus s_5 \ldots \oplus s_{2k+1}$, if n is odd; or

$A_2 = s_1 \oplus s_3 \oplus s_5 \ldots \oplus s_{2k-1}$, if n is even.

1.3 Concatenate A_1 and A_2 to get A.

2 Embed S into P by a certain data hiding technique to get a stego-PDF P'.

3 Embed A into P' by the following way.

3.1 Transform A into a bitstream $a_1a_2\ldots a_{16}$ with $a_i = 0$ or 1 where i is from 1 to

16.

3.2 Replace the last digit of the generation numbers of objects 1 through 16 in the cross-reference table by a_1, a_2, …and a_{16}, respectively.

3.3 Replace all generation numbers of object i in P' by the modified generation numbers, where i is from 1 to 16.

4 Take the resulting stego-PDF as the desired output P''.

The second part of the proposed authentication process is extracting the authentication signal and verifying the secret message for fidelity. Its flowchart is illustrated by Figure 4.3 and the detail is described below as an algorithm.

Algorithm 4.2. Extracting authentication signals and verifying the secret message in a stego-PDF.

Input: a stego-PDF P''.

Output: an authentication report R of the secret message in P''.

Steps:

1 Extract the secret message S in P''.

2 Apply exclusive-OR operations on all bytes of S to get an authentication signal A' by the following way.

2.1 Transform S into $S = s_1s_2s_3…s_n$, where s_1 through s_n are the bytes of S and n is the length of S in bytes.

2.2 Compute

$A_1 = s_2 \oplus s_4 \oplus s_6…\oplus s_{2k}$, where $k = n/2$;

$A_2 = s_1 \oplus s_3 \oplus s_5…\oplus s_{2k+1}$, if n is odd; or

$A_2 = s_1 \oplus s_3 \oplus s_5…\oplus s_{2k-1}$, if n is even.

2.3 Concatenate A_1 and A_2 to get A'.

3 Extract the embedded authentication signal A in P'' by the following way.

3.1 Find out the last digit of the generation numbers of objects 1 through 16 in the cross-reference table to get a_1, a_2, …and a_{16}, respectively, where $a_i = 0$ or 1 with i from 1 to 16.

3.2 Concatenate a_1 through a_{16} to get A.

4 Compare A and A'.

4.1 If $A = A'$, regard the authentication to be successful and mark it so in R.

4.2 Else, $A \neq A'$, decide that the secret message has been tampered with by illicit users and mark it so in R.

4.4 Experimental Results

We use the same program which was described in Chapter 3 for secret transmission to implement the authentication of the secret message in a stego-PDF. In Figure 4.4, we embed a secret message and its authentication signal into the cover PDF. In Figure 4.5, we extract the secret message first and then authenticate it to get the result of successful authentication. Figures 4.6 and 4.7 show the cover PDF and the stego-PDF with the secret message and authentication signal embedded, respectively. Comparing the two figures, no change can be seen on the display of the PDF document. If the secret message has been tampered with by illicit users, we will extract the wrong secret message. The result of the authentication is shown in Figure 4.8. It warns us that the secret message is not believable.

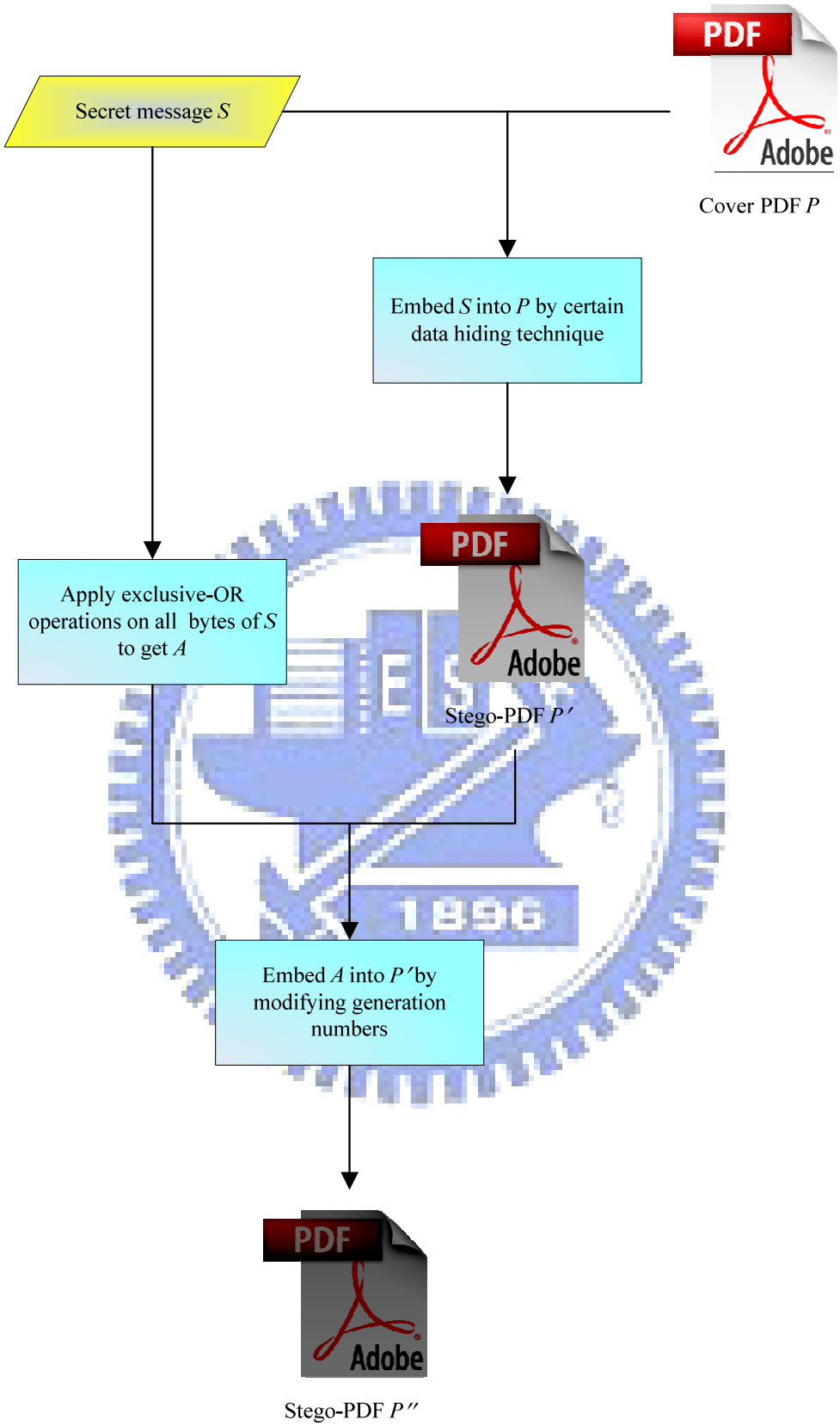


Figure 4.2 The flowchart of embedding authentication signals.

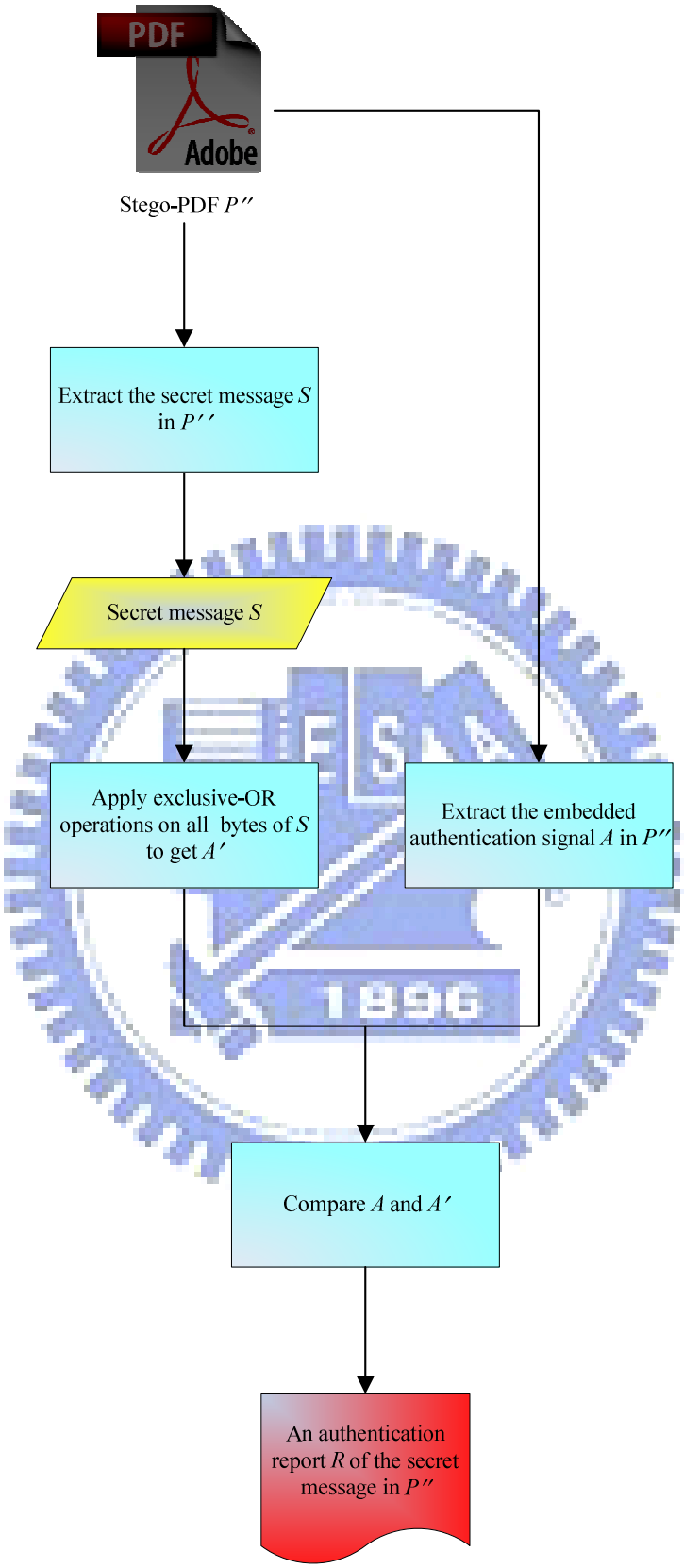


Figure 4.3 The flowchart of the authentication processes.

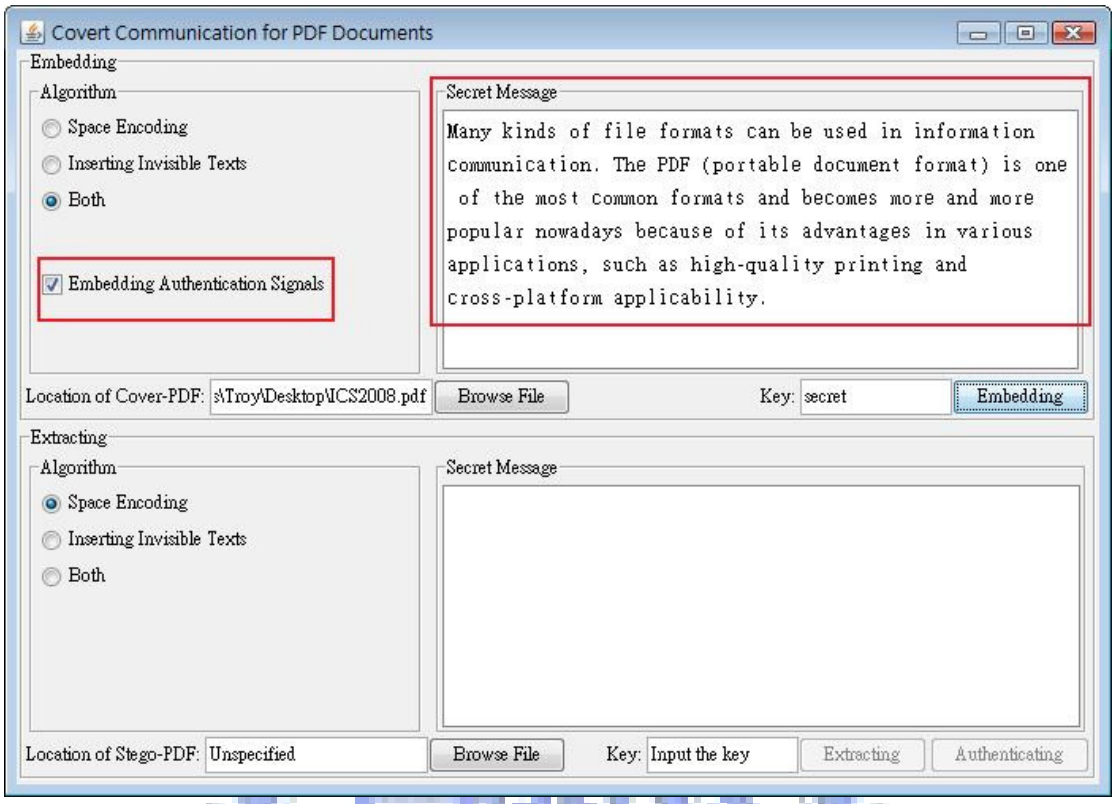


Figure 4.4 The window of the user interface for secret authentication

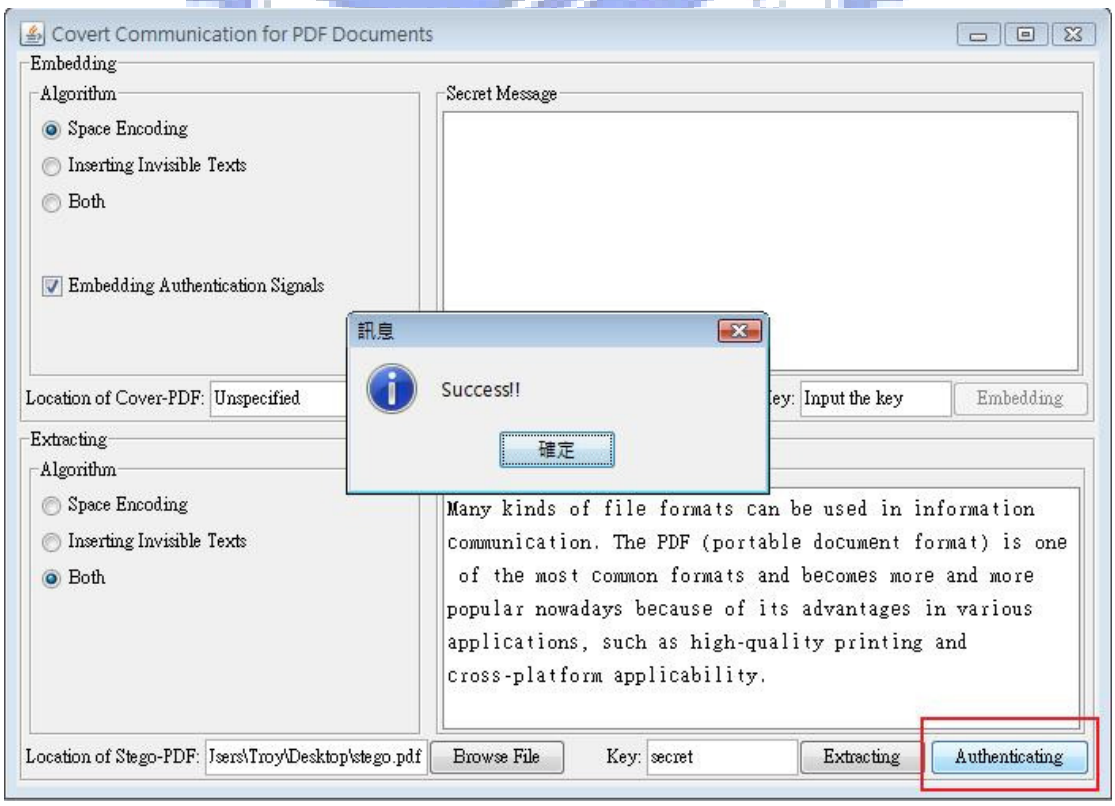
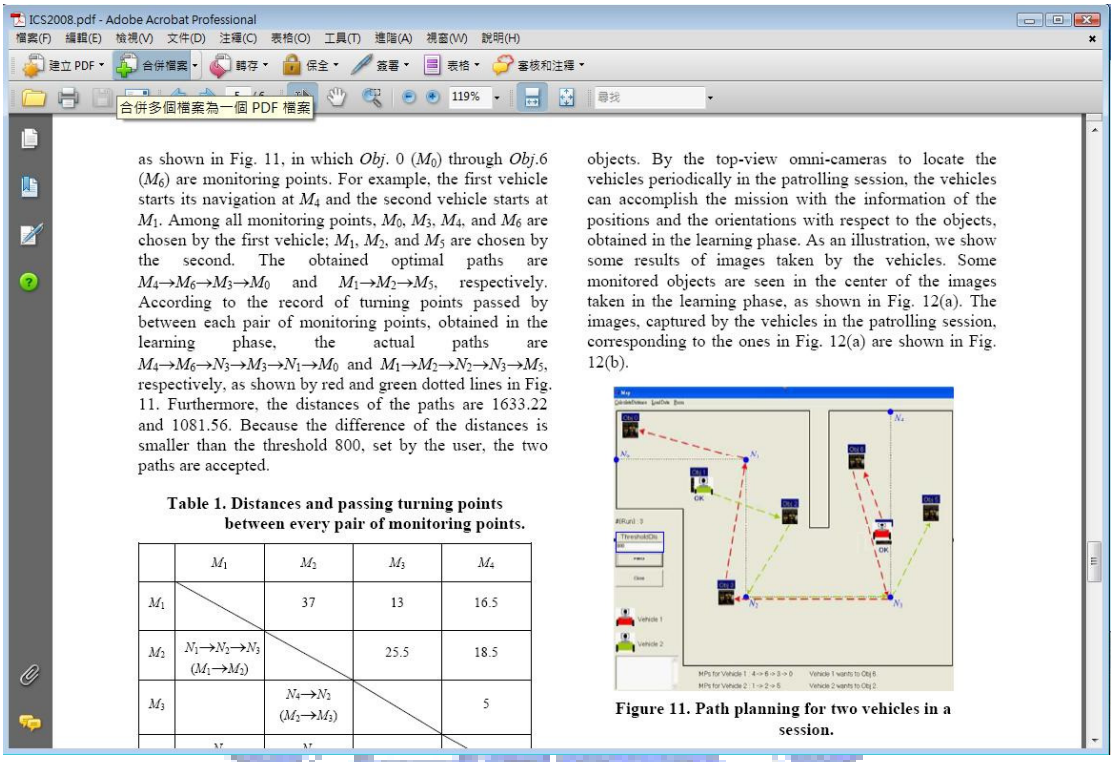


Figure 4.5 The result of successful authentication



as shown in Fig. 11, in which *Obj*. 0 (M_0) through *Obj*.6 (M_6) are monitoring points. For example, the first vehicle starts its navigation at M_4 and the second vehicle starts at M_1. Among all monitoring points, M_0, M_3, M_4, and M_6 are chosen by the first vehicle; M_1, M_2, and M_5 are chosen by the second. The obtained optimal paths are $M_4 \rightarrow M_6 \rightarrow M_3 \rightarrow M_0$ and $M_1 \rightarrow M_2 \rightarrow M_5$, respectively. According to the record of turning points passed by between each pair of monitoring points, obtained in the learning phase, the actual paths are $M_4 \rightarrow M_6 \rightarrow N_3 \rightarrow M_3 \rightarrow N_1 \rightarrow M_0$ and $M_1 \rightarrow M_2 \rightarrow N_2 \rightarrow N_3 \rightarrow M_5$, respectively, as shown by red and green dotted lines in Fig. 11. Furthermore, the distances of the paths are 1633.22 and 1081.56. Because the difference of the distances is smaller than the threshold 800, set by the user, the two paths are accepted.

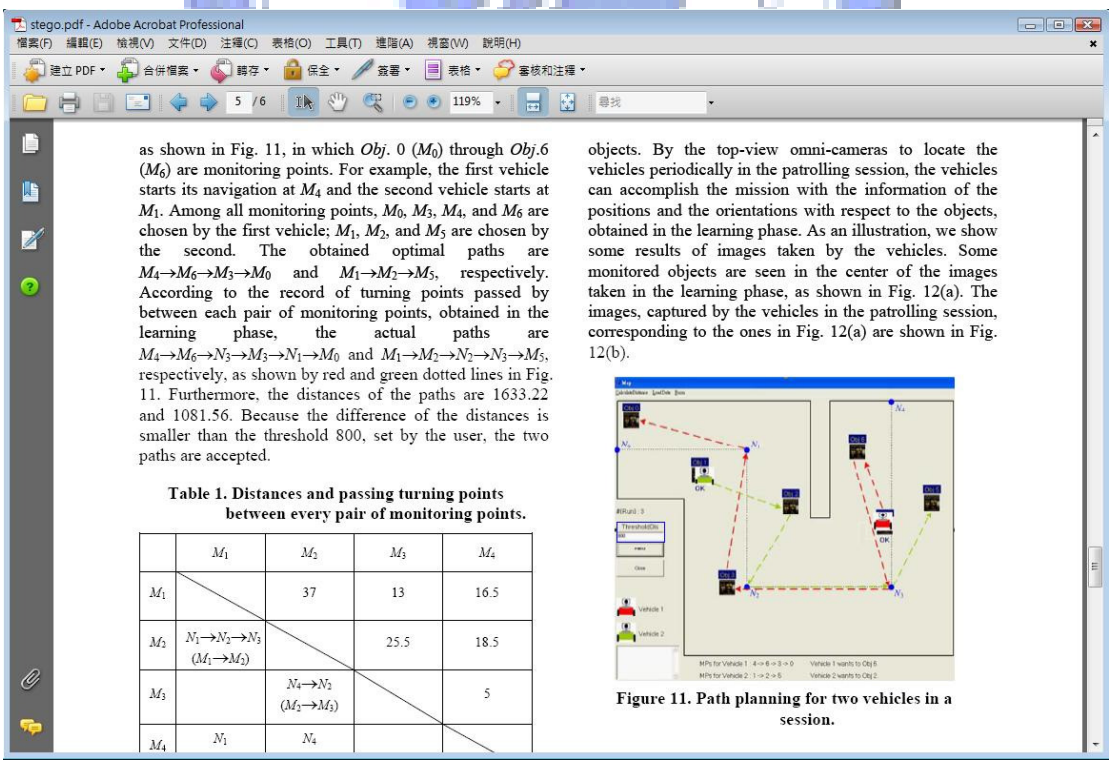
Table 1. Distances and passing turning points between every pair of monitoring points.

	M_1	M_2	M_3	M_4
M_1		37	13	16.5
M_2	$N_1 \rightarrow N_2 \rightarrow N_3$ ($M_1 \rightarrow M_2$)		25.5	18.5
M_3		$N_4 \rightarrow N_2$ ($M_2 \rightarrow M_3$)		5

objects. By the top-view omni-cameras to locate the vehicles periodically in the patrolling session, the vehicles can accomplish the mission with the information of the positions and the orientations with respect to the objects, obtained in the learning phase. As an illustration, we show some results of images taken by the vehicles. Some monitored objects are seen in the center of the images taken in the learning phase, as shown in Fig. 12(a). The images, captured by the vehicles in the patrolling session, corresponding to the ones in Fig. 12(a) are shown in Fig. 12(b).

Figure 11. Path planning for two vehicles in a session.

Figure 4.6 The cover PDF.



as shown in Fig. 11, in which *Obj*. 0 (M_0) through *Obj*.6 (M_6) are monitoring points. For example, the first vehicle starts its navigation at M_4 and the second vehicle starts at M_1. Among all monitoring points, M_0, M_3, M_4, and M_6 are chosen by the first vehicle; M_1, M_2, and M_5 are chosen by the second. The obtained optimal paths are $M_4 \rightarrow M_6 \rightarrow M_3 \rightarrow M_0$ and $M_1 \rightarrow M_2 \rightarrow M_5$, respectively. According to the record of turning points passed by between each pair of monitoring points, obtained in the learning phase, the actual paths are $M_4 \rightarrow M_6 \rightarrow N_3 \rightarrow M_3 \rightarrow N_1 \rightarrow M_0$ and $M_1 \rightarrow M_2 \rightarrow N_2 \rightarrow N_3 \rightarrow M_5$, respectively, as shown by red and green dotted lines in Fig. 11. Furthermore, the distances of the paths are 1633.22 and 1081.56. Because the difference of the distances is smaller than the threshold 800, set by the user, the two paths are accepted.

Table 1. Distances and passing turning points between every pair of monitoring points.

	M_1	M_2	M_3	M_4
M_1		37	13	16.5
M_2	$N_1 \rightarrow N_2 \rightarrow N_3$ ($M_1 \rightarrow M_2$)		25.5	18.5
M_3		$N_4 \rightarrow N_2$ ($M_2 \rightarrow M_3$)		5
M_4	N_1	N_4		

objects. By the top-view omni-cameras to locate the vehicles periodically in the patrolling session, the vehicles can accomplish the mission with the information of the positions and the orientations with respect to the objects, obtained in the learning phase. As an illustration, we show some results of images taken by the vehicles. Some monitored objects are seen in the center of the images taken in the learning phase, as shown in Fig. 12(a). The images, captured by the vehicles in the patrolling session, corresponding to the ones in Fig. 12(a) are shown in Fig. 12(b).

Figure 11. Path planning for two vehicles in a session.

Figure 4.7 The stego-PDF.

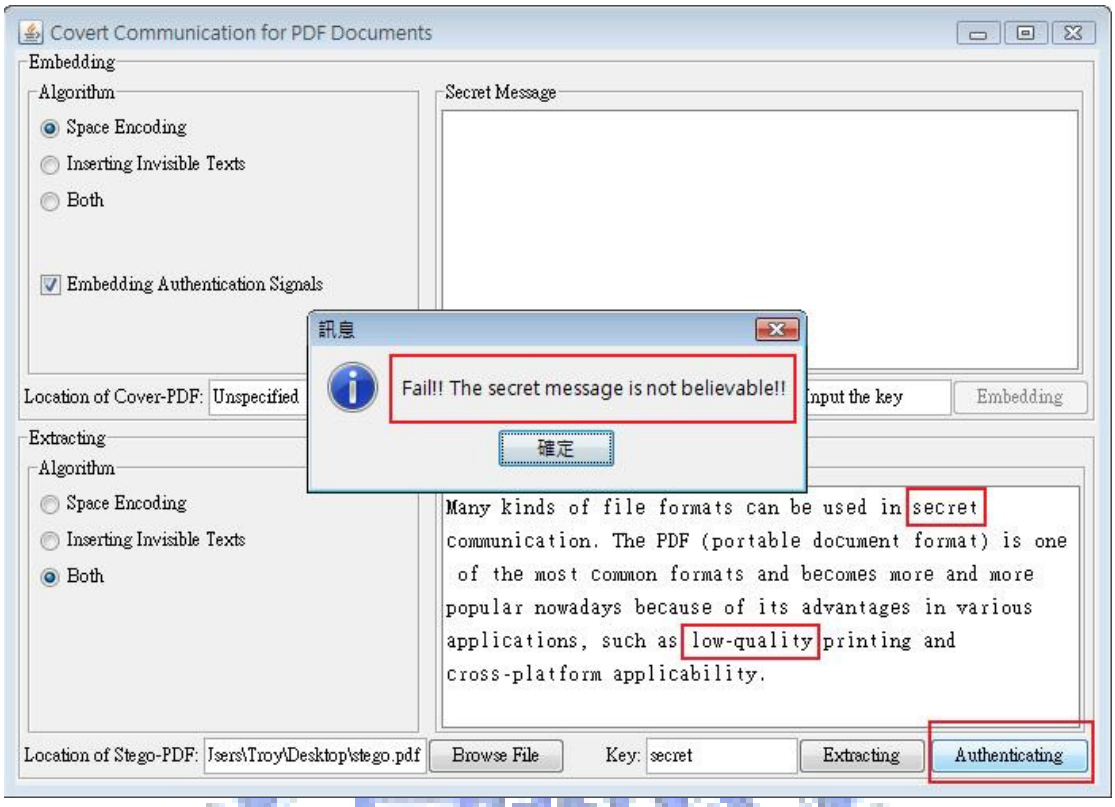


Figure 4.8 The result of failure authentication

4.5 Discussions and Summary

We have proposed a method for authentication of secret messages for fidelity verification in PDF files in this chapter. We generate authentication signals by applying exclusive-OR operations on each byte of the secret message and embed it into the secret-embedded PDF file by modifying generation numbers. It has no influence on the display of the PDF file. We use two bytes to embed the authentication signal in this study; however, if the secret message is long, we can generate more bytes of the authentication signals for secret authentication.

If we want to authenticate the secret message extracted from a stego-PDF, we just need to compare the authentication signal embedded in the stego-PDF and which is generated from the extracted secret message. Then we can know the result. The idea has been proved by our experiments.

Chapter 5
Secret Sharing via PDF Documents by Data Hiding Techniques

5.1 Introduction

Secret sharing is an interesting subject. For example, suppose that a treasure map is divided into several parts and distributed to some participants. Only when all the participants bring their partial maps together can the complete treasure map be recovered so that they can go to find the treasure. This way can prevent someone from finding and taking the treasure alone. Although a treasure map may only occur in stories, many things in the real word nowadays are just like the treasure map which needs protections, for example, certain copyrighted products which are not devised by single persons but a group of people in cooperation.

Many secret sharing methods have been proposed for different kinds of digital formats [8-10]. Because the PDF has become a very popular file format nowadays, many research papers and publications are saved as PDF documents. In addition to being used as cover files, the PDF document can be treated as a secret file to share as well.

In this chapter, the proposed method of secret sharing via PDF documents is described. An overview of the proposed method is stated in Section 5.2. And the proposed secret sharing and recovery algorithms are described in Sections 5.3 and 5.4, respectively. In Section 5.5, some experimental results of the proposed method is shown. Finally, a summary of the proposed method and some discussions are given in

Section 5.6.

5.2 Overview of Proposed Method of Secret Sharing via PDF Documents

In this section, the idea of the proposed method of secret sharing via PDF documents by data hiding techniques is described. The secret which we want to share is a PDF document, and the size of the secret PDF document is big in general. We use the data hiding technique based on the scheme of inserting invisible texts into cover PDFs, which is discussed in Section 3.2.2, to implement the proposed method.

The general idea of the proposed method is illustrated in Figure 5.1. We have a secret PDF document and select n cover PDFs randomly, denoted as P_1 through P_n, respectively. After applying the proposed secret sharing process, we transform them into n stego-PDFs which are regarded as *shares* and denoted as P_1' through P_n', respectively. Then we distribute the n shares to a group of participants of the same number, each participant with a share.

When the participants want to recover the secret PDF document, they should bring their shares all together. After applying the proposed secret recovery process, they can get the original secret PDF document.

From each cover PDF, we extract a portion of data with the same size as that of the secret PDF file. Every portion of data is regarded as a *preliminary share*. Let the length of the secret PDF document in bytes be denoted as l. The idea of extracting data from each cover PDF to prepare the preliminary share is described below.

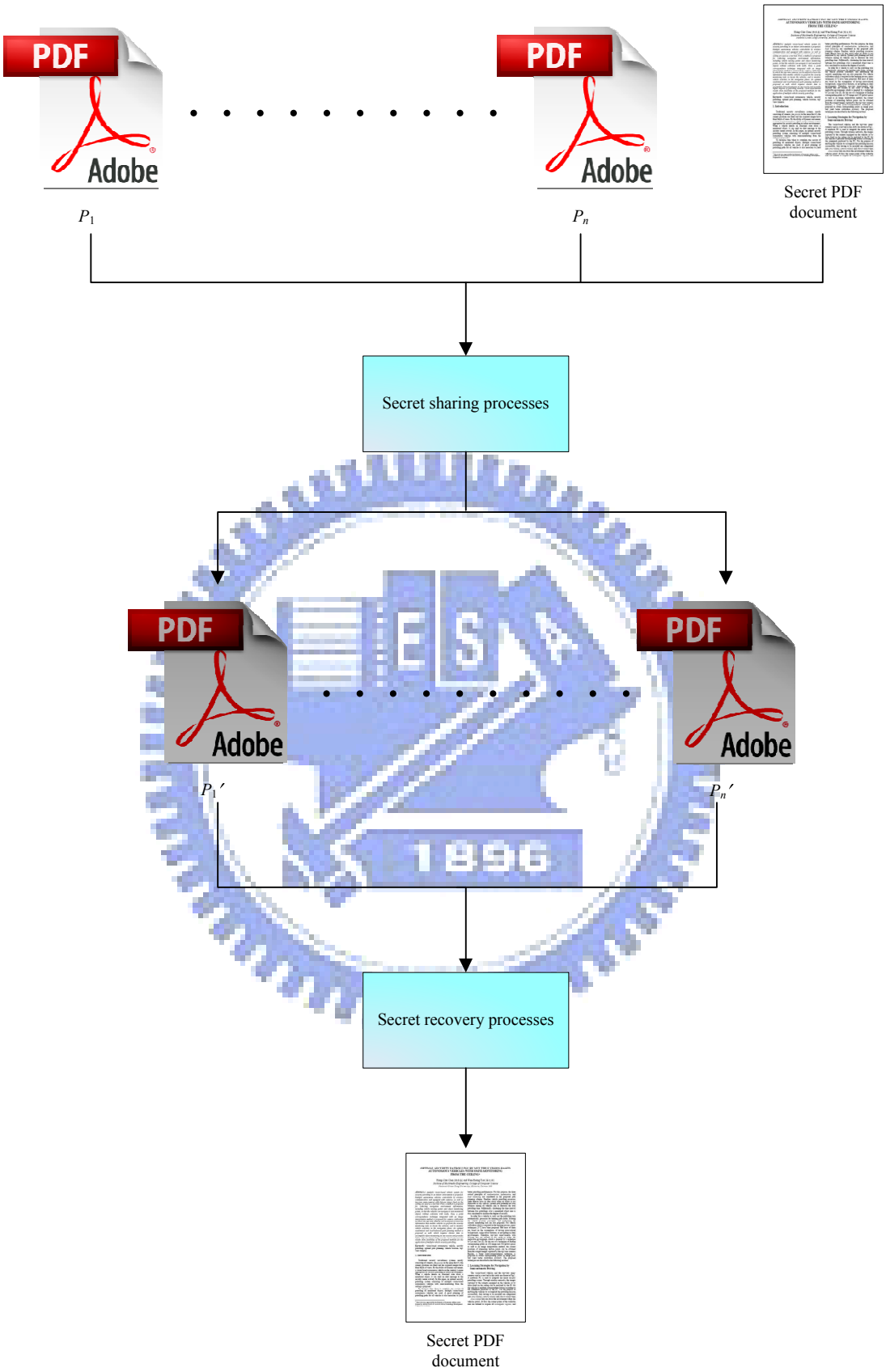


Figure 5.1 Illustration of proposed method of secret sharing via PDF documents.

1. If the size of the cover PDF is larger than or equal to that of the secret PDF document, we extract data whose length is l from the beginning of the cover PDF.
2. If the size of the cover PDF is smaller than that of the secret PDF document, we extract data from the beginning of the cover PDF to its end and repeat this operation, until the total length of the extracted data is l.
3. As mentioned in Section 3.2.2, after embedding data into an indirect object by the data hiding technique of inserting invisible texts, we need to update the cross-reference table and the trailer in the cover PDF. It means that the indirect object in the stego-PDF where we embed data, the cross-reference table, and the trailer are not the same as they are in the cover PDF. Because we will use the data hiding technique of inserting invisible texts in the later processes to embed the intermediate secret message, when we extract data from each cover PDF to get preliminary shares, we skip the indirect object with data embedded, the cross-reference table, and the trailer to guarantee that when we extract the preliminary shares from the stego-PDFs, they are still the same as those extracted from the cover PDFs, so that the recovered PDF document is correct.

After the above process, we can get n preliminary shares with the same size. We then use them and the secret PDF document to generate the *intermediate secret message* by using exclusive-OR and coincidence operators. Which operator should be used is decided by a key K generated from the user keys selected and kept by the participants, respectively. Let the user key of participant P_i be denoted as K_i, where i is from 1 to n. Then, we generate K as $K = K_1 \oplus K_2 \ldots \oplus K_n$.

We know that $p \odot q = -(p \oplus q)$, where "$\oplus$" is the exclusive-OR operator and "$\odot$" is the coincidence operator. Because "$\oplus$" is commutative, $p \odot q = -(p \oplus q) = -(q \oplus p) = q \odot p$, "$\odot$" is also commutative. So we can use exclusive-OR and coincidence operators to generate the intermediate secret message S'.

Denote each preliminary share as $S_i = s_{i1}s_{i2}\ldots s_{il}$ where i is from 1 to n and s_{i1} through s_{il} are the bytes of S_i, and the intermediate secret message as $S' = s'_1 s'_2 \ldots s'_l$ where s'_1 through s'_l are the bytes of S'. Also, denote the secret PDF document as $S = s_1 s_2 \ldots s_l$ where s_1 through s_l are the bytes of S. Also, let $K = k_1 k_2 \ldots k_h$ where h is the length of K in bits and k_1 through k_h are the bits of K. Then, we generate S' by the following way:

$s'_1 = s_1 \bigcirc_1 s_{11} \bigcirc_2 s_{21} \bigcirc_3 s_{31} \ldots \bigcirc_n s_{n1}$

$s'_2 = s_2 \bigcirc_2 s_{12} \bigcirc_3 s_{22} \bigcirc_4 s_{32} \ldots \bigcirc_{n+1} s_{n2}$

…

$s'_l = s_l \bigcirc_l s_{1l} \bigcirc_{l+1} s_{2l} \bigcirc_{l+2} s_{3l} \ldots \bigcirc_{n+l-1} s_{nl}$,

where $\bigcirc_j$ with j is from 1 to $n+l-1$ is either "$\oplus$" or "$\odot$," determined by the following rule:

$\bigcirc_j = \oplus$, if the value of $k_{(j \bmod h)}$ is 1; or

$\bigcirc_j = \odot$, if the value of $k_{(j \bmod h)}$ is 0.

After we get the intermediate secret message S', we separate S' into n parts with the length of each part being l/n in bytes. Then, we embed each part into one of P_1 through P_n, respectively, by the scheme of inserting invisible texts which is discussed in Section 3.2.2, to get the stego-PDF's P_1' through P_n', respectively. Figure 5.2 shows an illustration of this secret sharing process.

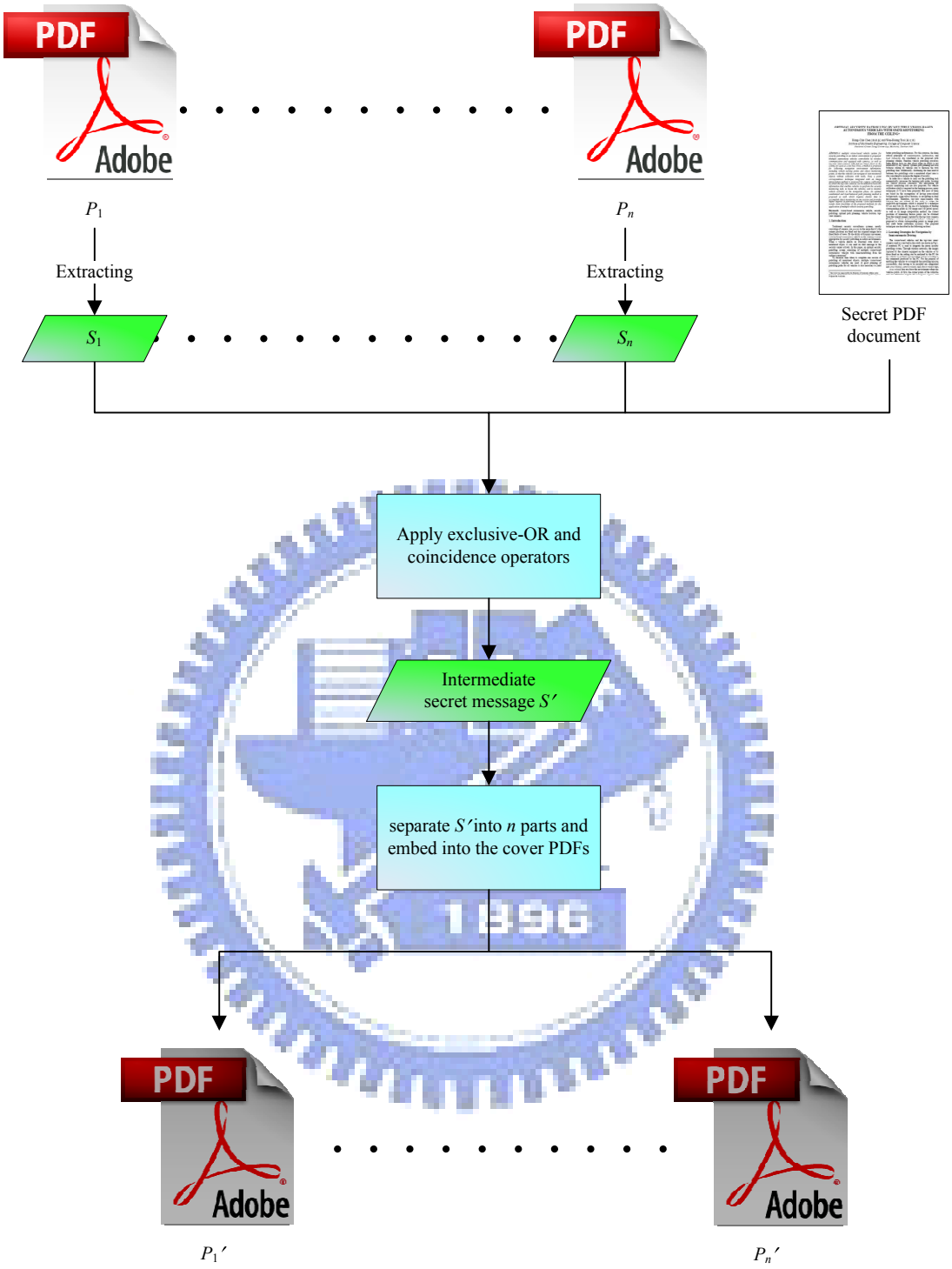


Figure 5.2 Illustration of proposed secret sharing process.

If we want to recover the secret PDF document, the participants should bring the n stego-PDFs all together. First, we extract n parts of the intermediate secret message from the stego-PDFs and concatenate them to get S' with length l. Then, we get n

preliminary shares from the stego-PDFs by a reverse process of the way mentioned in Section 5.2.1 and generate K as $K = K_1 \oplus K_2 \ldots \oplus K_n$. Denote each preliminary share as $S_i = s_{i1}s_{i2}\ldots s_{il}$ where i is from 1 to n and s_{i1} through s_{il} are the bytes of S_i, and the intermediate secret message as $S' = s'_1 s'_2 \ldots s'_l$ where s'_1 through s'_l are the bytes of S'. Also denote the recovered secret PDF document as $S = s_1 s_2 \ldots s_l$ where s_1 through s_l are the bytes of S. Let $K = k_1 k_2 \ldots k_h$ where h is the length of K in bits and k_1 through k_h are the bits of K. Then, we recover S by the following way:

$s_1 = s'_1 \bigcirc_1 s_{11} \bigcirc_2 s_{21} \bigcirc_3 s_{31} \ldots \bigcirc_n s_{n1}$

$s_2 = s'_2 \bigcirc_2 s_{12} \bigcirc_3 s_{22} \bigcirc_4 s_{32} \ldots \bigcirc_{n+1} s_{n2}$

…

$s_l = s'_l \bigcirc_l s_{1l} \bigcirc_{l+1} s_{2l} \bigcirc_{l+2} s_{3l} \ldots \bigcirc_{n+l-1} s_{nl}$,

where $\bigcirc_j$ with j is from 1 to $n+l-1$ is either "$\oplus$" or "$\odot$," determined by the rule:

$\bigcirc_j = \oplus$, if the value of $k_{(j \bmod h)}$ is 1; or

$\bigcirc_j = \odot$, if the value of $k_{(j \bmod h)}$ is 0.

Finally, we recover the secret PDF document S. An illustration of the proposed secret recovery process is shown in Figure 5.3.

5.3 Proposed Secret Sharing Algorithm

The idea of the proposed secret PDF document sharing has been described in the last section and the detail is described below as an algorithm.

Algorithm 5.1. Sharing a secret PDF document.

Input: a secret PDF document S and n cover PDFs P_1, P_2,…, P_n with n user-selected keys K_1, K_2, …, K_n.

Output: n stego-PDFs P_1', P_2',…, P_n'.

Steps:

1 Extract S_1, S_2,…, S_n from P_1, P_2,…, P_n, respectively, by the way mentioned in Section 5.2.1 to get $S_i = s_{i1}s_{i2}\ldots s_{il}$, where i is from 1 to n, l is the length of S and s_{i1} through s_{il} are the bytes of S_i.

2 Transform S into $S = s_1s_2s_3\ldots s_l$, where s_1 through s_l are the bytes of S.

3 Generate the key $K = K_1 \oplus K_2 \ldots \oplus K_n$.

4 Generate the intermediate secret message $S' = s'_1 s'_2 \ldots s'_l$, where s'_1 through s'_l are the bytes of S', by the following way:

$s'_1 = s_1 \bigcirc_1 s_{11} \bigcirc_2 s_{21} \bigcirc_3 s_{31} \ldots \bigcirc_n s_{n1}$

$s'_2 = s_2 \bigcirc_2 s_{12} \bigcirc_3 s_{22} \bigcirc_4 s_{32} \ldots \bigcirc_{n+1} s_{n2}$

…

$s'_l = s_l \bigcirc_l s_{1l} \bigcirc_{l+1} s_{2l} \bigcirc_{l+2} s_{3l} \ldots \bigcirc_{n+l-1} s_{nl}$,

where $\bigcirc_j$ with j is from 1 to $n+l-1$ is either "⊕" or "⊙," determined by:

$\bigcirc_j =$ ⊕, if the value of $k_{(j \bmod h)}$ is 1; or

$\bigcirc_j =$ ⊙, if the value of $k_{(j \bmod h)}$ is 0.

5 Separate S' into n parts S'_1, S'_2,…, S'_n with the length of each part being l/n in bytes.

6 Embed S'_i into P_i by inserting invisible texts to get P_i', where i is from 1 to n.

5.4 Proposed Secret Recovery Algorithm

The detail of the proposed secret recovery processes via PDF documents is described below as an algorithm.

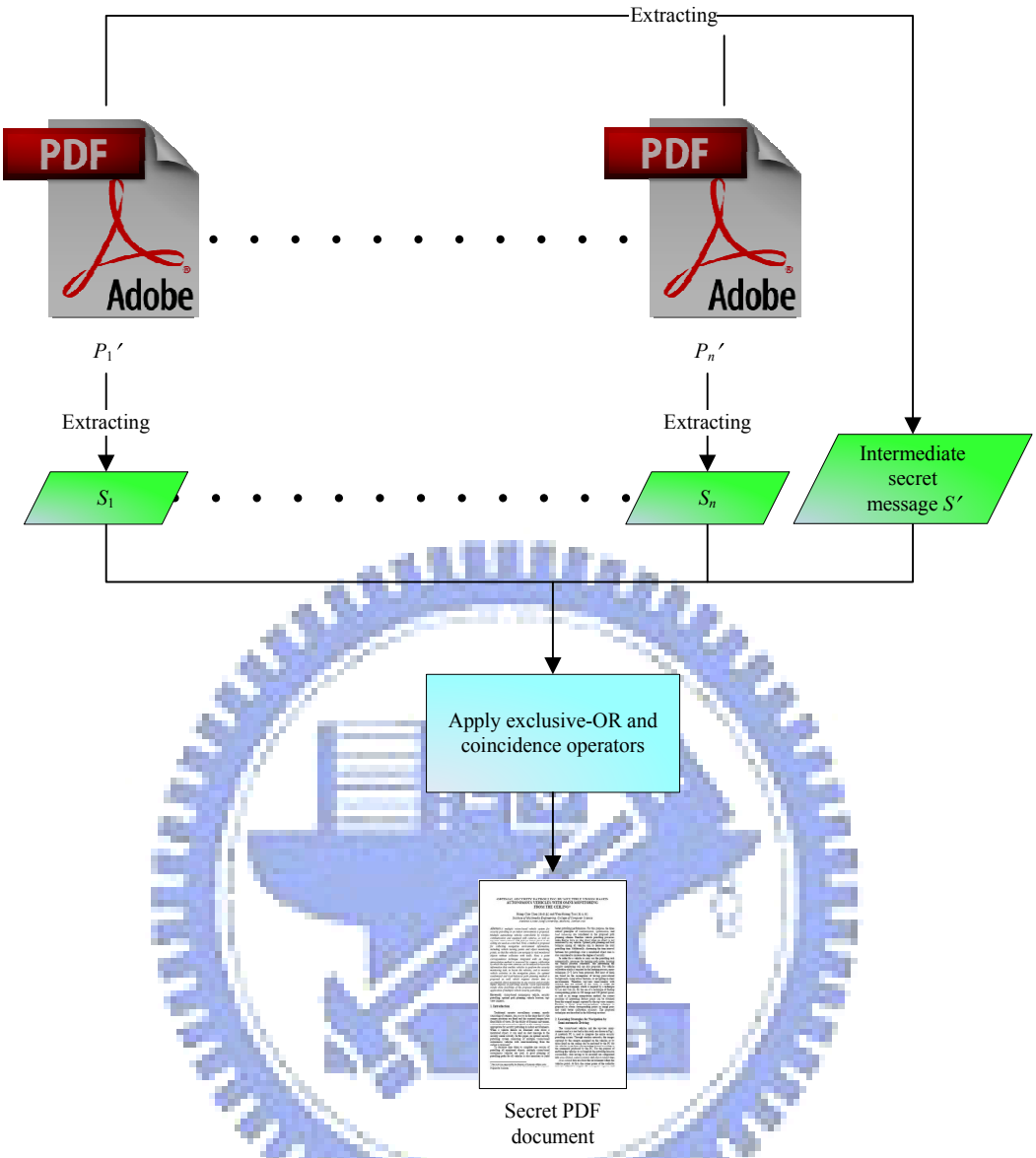


Figure 5.3 Illustration of proposed secret recovery process.

Algorithm 5.2. Recovering a secret PDF document.

Input: n stego-PDFs P_1', P_2',…, P_n' with n user-selected keys K_1, K_2, …, K_n.

Output: a secret PDF document S.

Steps:

1 Extract the embedded data S'_1, S'_2,…, S'_n from P_1', P_2',…, P_n', respectively.

2 Concatenate S'_1 through S'_n to get intermediate secret message S' with length l.

3 Transform S' into $S' = s'_1 s'_2 \ldots s'_l$, where s'_1 through s'_l are the bytes of S'.

4 Extract S_1, S_2,…, S_n from P_1', P_2',…, P_n', respectively, by the way mentioned in Section 5.2.1 to get $S_i = s_{i1}s_{i2}\ldots s_{il}$, where i is from 1 to n and s_{i1} through s_{il} are the bytes of S_i.

5 Generate the key $K = K_1 \oplus K_2 \ldots \oplus K_n$.

6 Recover $S = s_1 s_2 \ldots s_l$ where s_1 through s_l are the bytes of S by the following way.

$s_1 = s'_1 \bigcirc_1 s_{11} \bigcirc_2 s_{21} \bigcirc_3 s_{31} \ldots \bigcirc_n s_{n1}$

$s_2 = s'_2 \bigcirc_2 s_{12} \bigcirc_3 s_{22} \bigcirc_4 s_{32} \ldots \bigcirc_{n+1} s_{n2}$

…

$s_l = s'_l \bigcirc_l s_{1l} \bigcirc_{l+1} s_{2l} \bigcirc_{l+2} s_{3l} \ldots \bigcirc_{n+l-1} s_{nl}$,

where $\bigcirc_j$ with j is from 1 to $n+l-1$ is either "⊕" or "⊙," determined by:

$\bigcirc_j = \oplus$, if the value of $k_{(j \bmod h)}$ is 1; or

$\bigcirc_j = \odot$, if the value of $k_{(j \bmod h)}$ is 0.

5.5 Experimental Results

In order to implement the proposed method for secret sharing via PDF documents, we designed a user interface for the program we have written in the language of Java. It supports both the secret sharing and recovery functions. In one of our experiments, we used three cover PDFs to share a secret PDF document.

Figure 5.4 shows the dialog window for keying in the number of participants in the secret sharing activity, We can decide the number of cover PDFs according to this number of participants. We use three cover PDFs in this experiment. Figure 5.5 shows the window of the user interface, and Figure 5.6 shows this window with a secret PDF document and three cover PDFs with user keys as input. The three cover PDFs are shown in Figures 5.7, 5.8 and 5.9, respectively, and the secret PDF document is

shown in Figure 5.10.

After the proposed secret sharing process was conducted, we get three stego-PDFs, as shown in Figures 5.11, 5.12 and 5.13, respectively. No change can be seen on the displays of the stego-PDF documents, compared with those of the original cover PDF. And then we can distribute them to the three participants, each participant with a share.

Figure 5.14 shows the window of the user interface with the three stego-PDFs and user keys as input to recover the secret PDF document. And the recovered PDF document is shown in Figure 5.15. We can find out that the recovered PDF document is just the same one as the original secret PDF document shown in Figure 5.10. As long as one of the shares or the user keys is incorrect, the recovered PDF document cannot be opened, as shown in Figure 5.16.

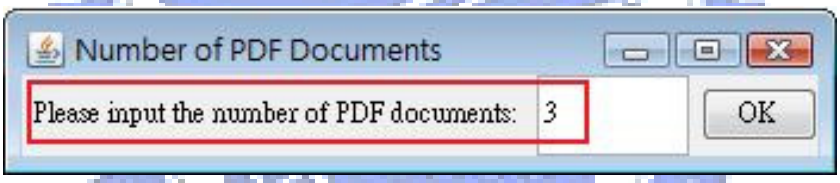


Figure 5.4 The window of inputting the number of PDF documents.

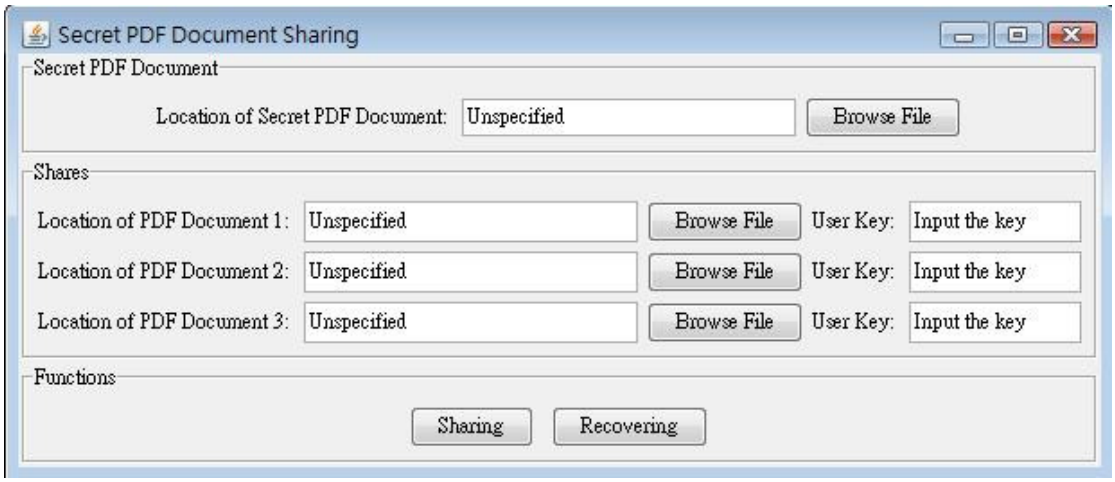


Figure 5.5 The window of the user interface.

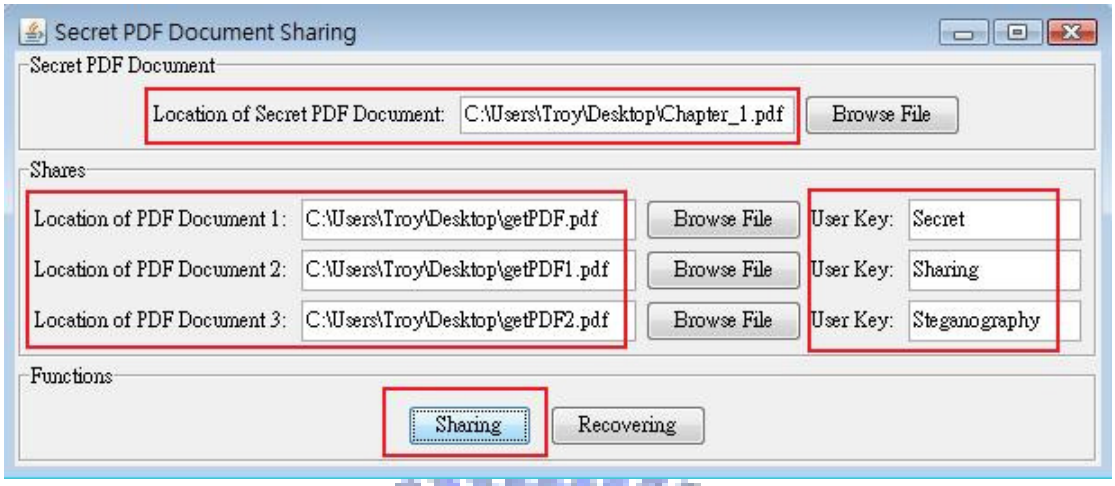


Figure 5.6 The window of the user interface with a secret PDF document and three cover PDFs with user keys as input

.

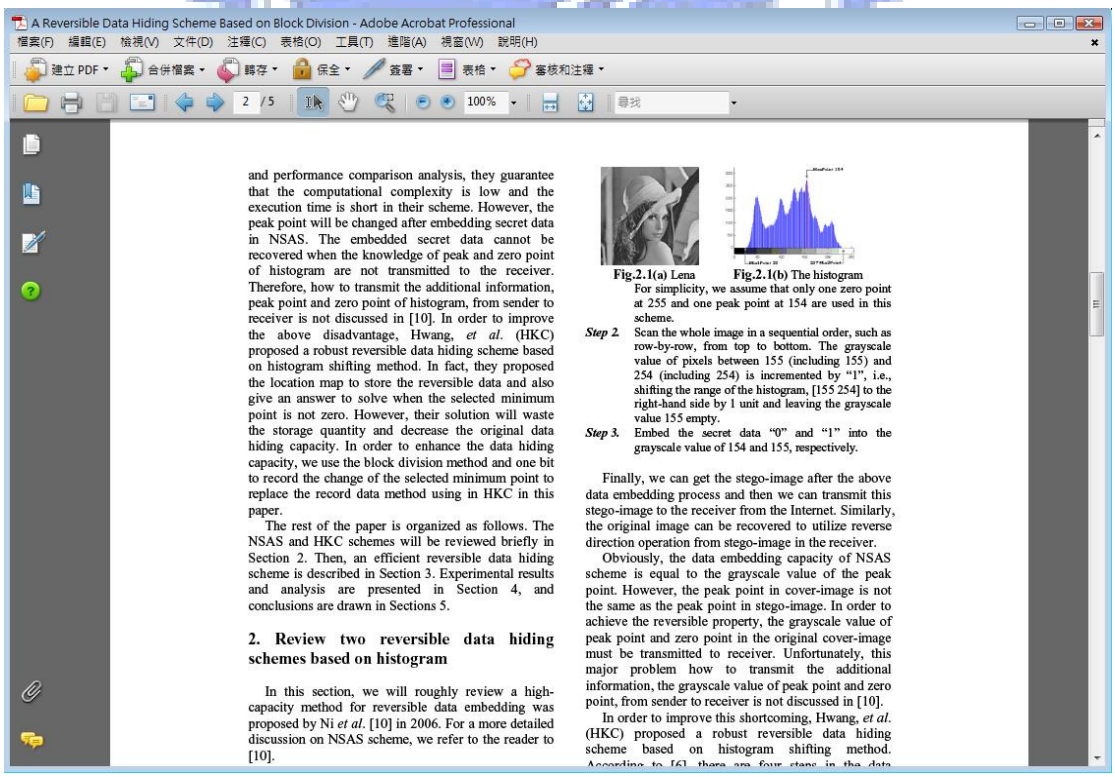


Figure 5.7 The first cover PDF

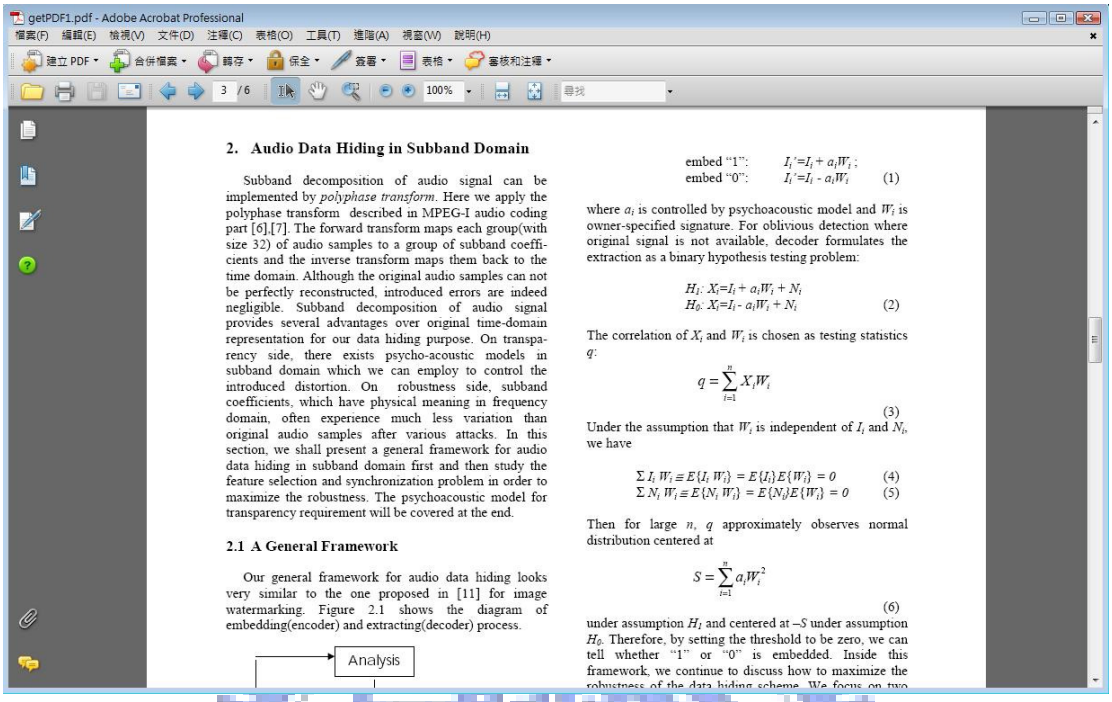


Figure 5.8 The second cover PDF

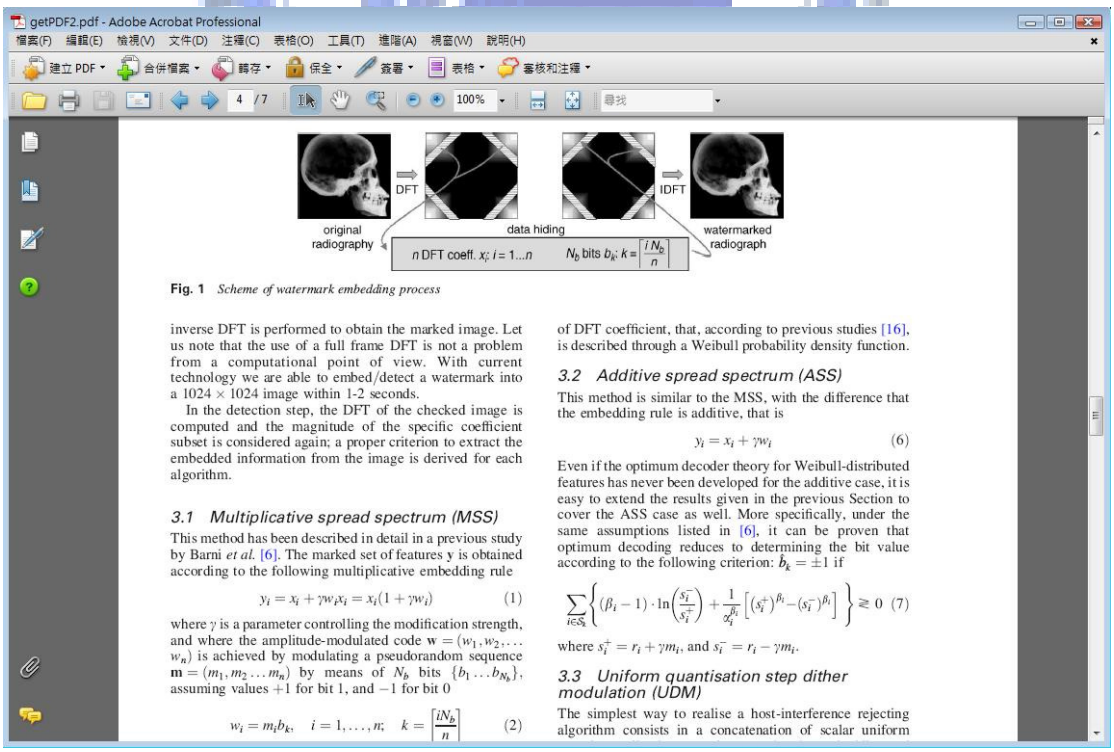


Figure 5.9 The third cover PDF

Chapter 1
Introduction

1.1 Motivation

Information communication plays a very important role in the development of science and technology. Initially, humans exchange information by talking to one another. Later, people write messages on paper and send them to others. Today, people can exchange information on the Internet without face-to-face meeting. The exchange process is time-saving and convenient, but we need to pay attention to the safety and accuracy of the information which we send.

Many kinds of file formats can be used in information communication. The PDF (portable document format) is one of the most common formats and becomes more and more popular nowadays because of its advantages in various applications, such as high-quality printing and cross-platform applicability. In addition to using PDF files directly to exchange information, it is also advantageous to use them as *cover media* to hide data for transmitting secret messages and other uses. Although PDF files are very popular, there are yet not many researches on data hiding in PDF files. It is desirable to have more kinds of data hiding techniques for various application purposes.

Furthermore, it is not safe for people to transmit secret messages on the Internet. The messages which people send may be intercepted and tampered with even if data hiding techniques are used. So it is better to design certain schemes for protections of

Figure 5.10 The secret PDF document.

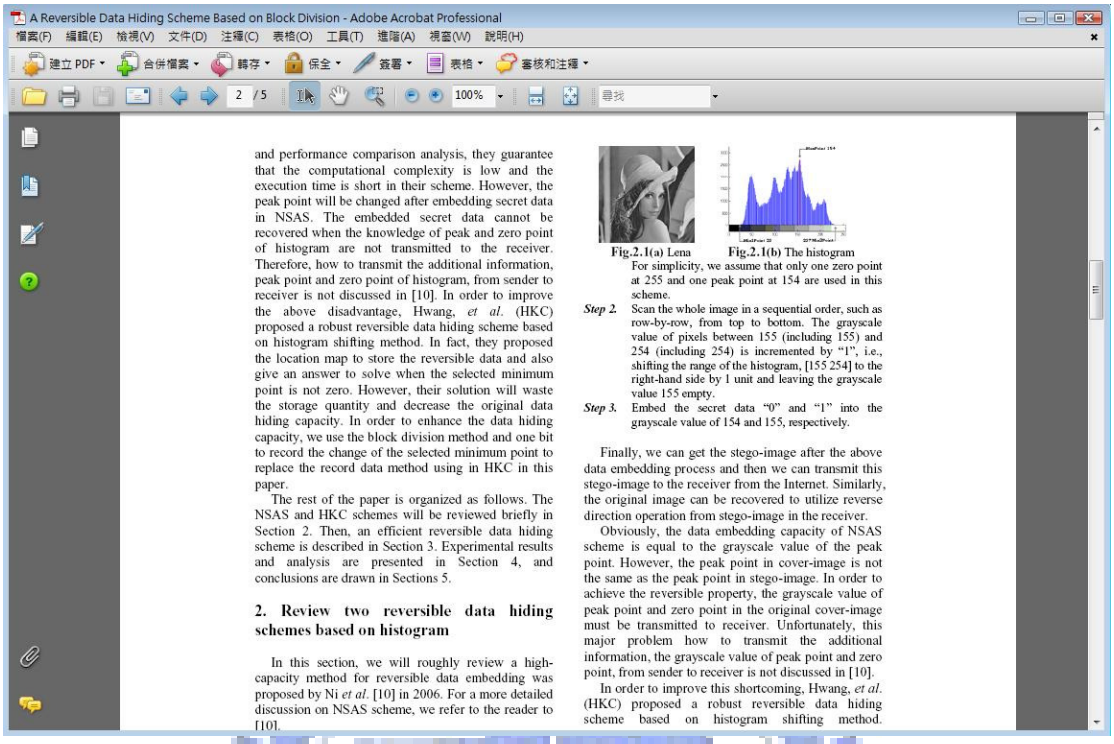


Figure 5.11 The first stego-PDF.

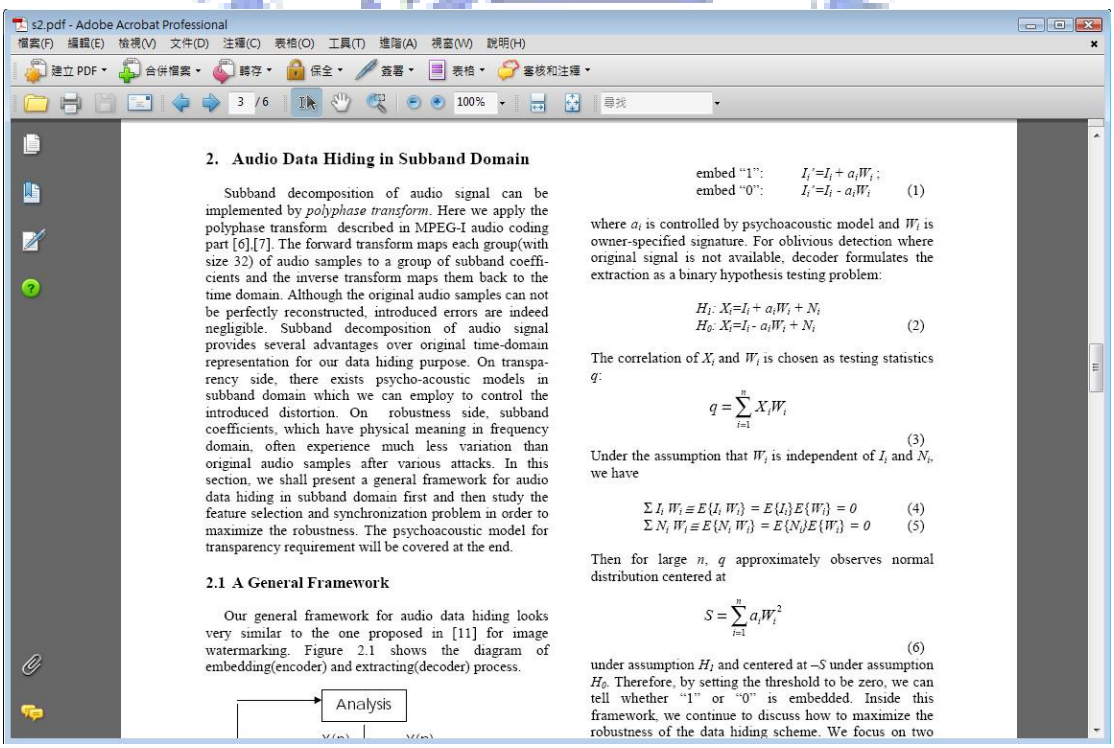


Figure 5.12 The second stego-PDF.

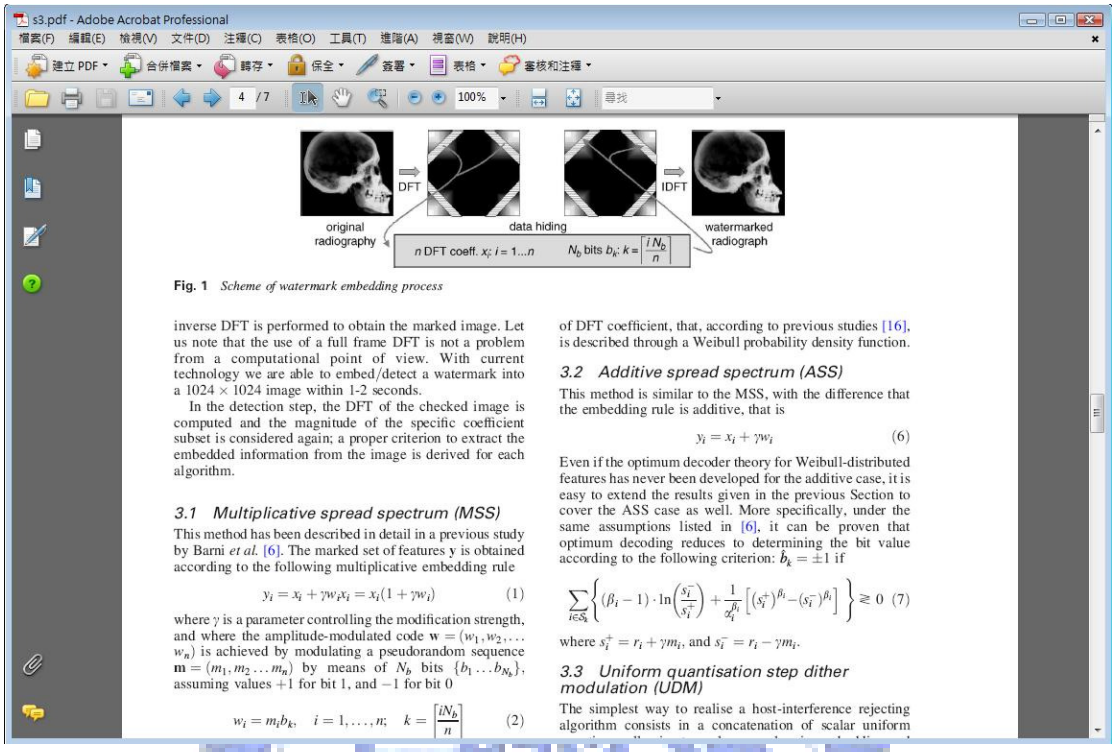


Fig. 1 *Scheme of watermark embedding process*

inverse DFT is performed to obtain the marked image. Let us note that the use of a full frame DFT is not a problem from a computational point of view. With current technology we are able to embed/detect a watermark into a 1024×1024 image within 1-2 seconds.

In the detection step, the DFT of the checked image is computed and the magnitude of the specific coefficient subset is considered again; a proper criterion to extract the embedded information from the image is derived for each algorithm.

3.1 Multiplicative spread spectrum (MSS)

This method has been described in detail in a previous study by Barni *et al.* [6]. The marked set of features **y** is obtained according to the following multiplicative embedding rule

$$y_i = x_i + \gamma w_i x_i = x_i(1 + \gamma w_i) \quad (1)$$

where γ is a parameter controlling the modification strength, and where the amplitude-modulated code $\mathbf{w} = (w_1, w_2, \ldots w_n)$ is achieved by modulating a pseudorandom sequence $\mathbf{m} = (m_1, m_2 \ldots m_n)$ by means of N_b bits $\{b_1 \ldots b_{N_b}\}$, assuming values $+1$ for bit 1, and -1 for bit 0

$$w_i = m_i b_k, \quad i = 1, \ldots, n; \quad k = \left\lfloor \frac{iN_b}{n} \right\rfloor \quad (2)$$

of DFT coefficient, that, according to previous studies [16], is described through a Weibull probability density function.

3.2 Additive spread spectrum (ASS)

This method is similar to the MSS, with the difference that the embedding rule is additive, that is

$$y_i = x_i + \gamma w_i \quad (6)$$

Even if the optimum decoder theory for Weibull-distributed features has never been developed for the additive case, it is easy to extend the results given in the previous Section to cover the ASS case as well. More specifically, under the same assumptions listed in [6], it can be proven that optimum decoding reduces to determining the bit value according to the following criterion: $\hat{b}_k = \pm 1$ if

$$\sum_{i \in S_k} \left\{ (\beta_i - 1) \cdot \ln\left(\frac{s_i^-}{s_i^+}\right) + \frac{1}{\alpha_i^{\beta_i}} \left[(s_i^+)^{\beta_i} - (s_i^-)^{\beta_i} \right] \right\} \gtrless 0 \quad (7)$$

where $s_i^+ = r_i + \gamma m_i$, and $s_i^- = r_i - \gamma m_i$.

3.3 Uniform quantisation step dither modulation (UDM)

The simplest way to realise a host-interference rejecting algorithm consists in a concatenation of scalar uniform

Figure 5.13 The third stego-PDF.

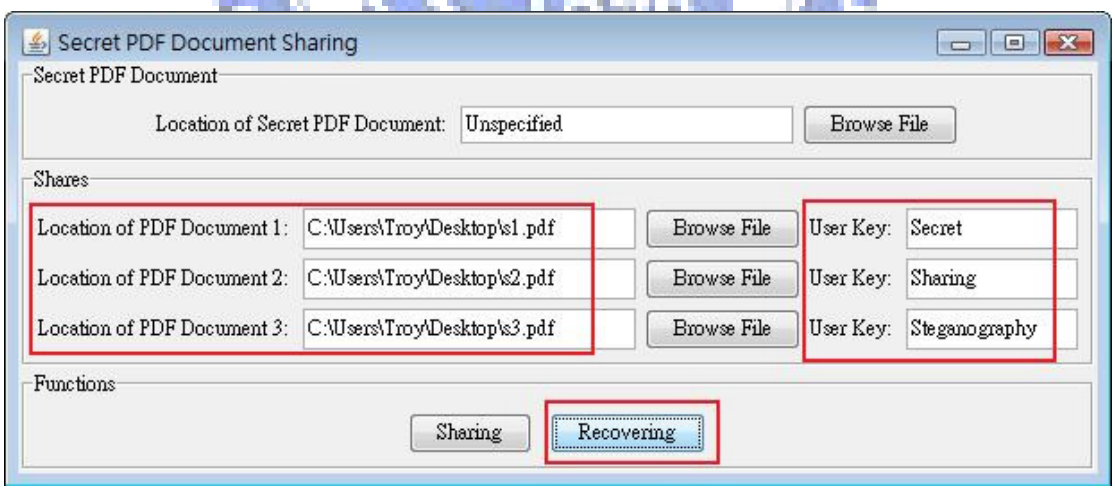


Figure 5.14 The window of the user interface with three stego-PDFs and user keys as input.

Chapter 1
Introduction

1.1 Motivation

Information communication plays a very important role in the development of science and technology. Initially, humans exchange information by talking to one another. Later, people write messages on paper and send them to others. Today, people can exchange information on the Internet without face-to-face meeting. The exchange process is time-saving and convenient, but we need to pay attention to the safety and accuracy of the information which we send.

Many kinds of file formats can be used in information communication. The PDF (portable document format) is one of the most common formats and becomes more and more popular nowadays because of its advantages in various applications, such as high-quality printing and cross-platform applicability. In addition to using PDF files directly to exchange information, it is also advantageous to use them as *cover media* to hide data for transmitting secret messages and other uses. Although PDF files are very popular, there are yet not many researches on data hiding in PDF files. It is desirable to have more kinds of data hiding techniques for various application purposes.

Furthermore, it is not safe for people to transmit secret messages on the Internet. The messages which people send may be intercepted and tampered with even if data hiding techniques are used. So it is better to design certain schemes for protections of

Figure 5.15 The recovered PDF document.

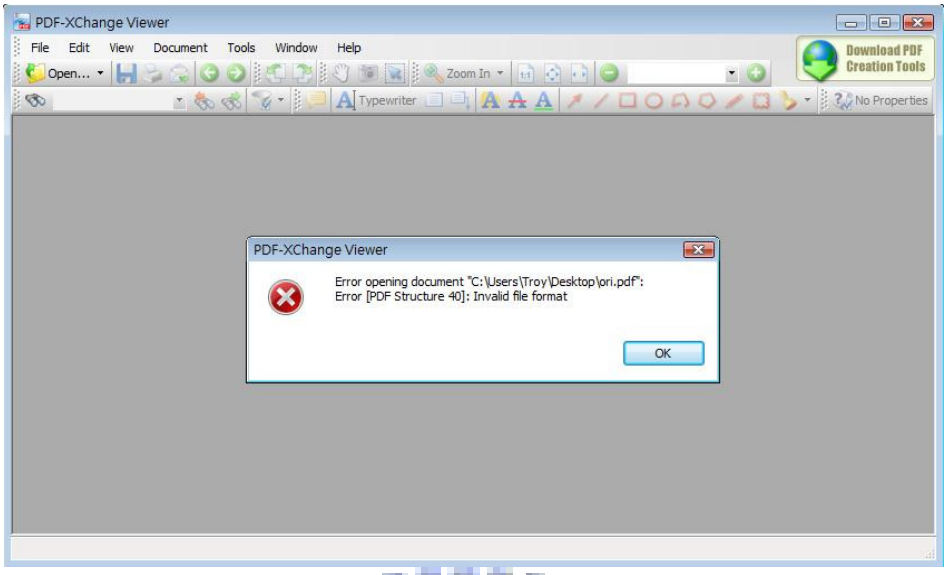


Figure 5.16 The incorrect secret PDF document.

5.6 Discussions and Summary

We have proposed a method for secret sharing via PDF documents in this chapter. We use exclusive-OR and coincidence operators to share the secret PDF document and embed the intermediate secret message into cover PDFs by the data hiding technique which is mentioned in Chapter 3. By the proposed secret sharing method, we can select cover PDFs and user keys randomly. After applying the proposed sharing method, we can get shares and distribute them to a group of participants of the same number, each participant with a share. Only when all the participants bring their shares with correct user keys together can the secret PDF document be recovered. So this way of secret keeping can prevent someone from occupying the secret PDF document alone. Even if an illicit user steals all the shares, he/she still cannot recover the secret PDF document without the correct user keys from other participants. The idea has been proved by our experiments. Compared with the cover PDFs, the shares created in our experiments showed to have no change on the displays of the PDF documents, and the recovered secret PDF document was intact.

Chapter 6
Conclusions and Suggestions for Future Works

6.1 Conclusions

In this study, we have proposed a data hiding method for PDF files as cover media. It is useful for some applications, such as secret transmission, document authentication, and secret sharing.

For secret transmission, we have proposed two data hiding techniques via PDF files. We can transmit a secret message by using one of them or both. The first is a technique based on the space coding scheme. Although the capacity of the embedded data is limited by the number of usable white-space characters in the cover PDF, the size of the cover-PDF will not change after embedding the secret message. The second is a technique of inserting invisible texts into a cover PDF. The capacity of the data which we can embed in a PDF file may be said to be unlimited, at the sacrifice of the resulting size of the stego-PDF file. No matter which technique, or both of them, are used to implement secret transmission, it has no influence on the display of the PDF file so that people will not be aware of the existence of the hidden data. Even if an illicit user knows the algorithm which we use to embed the secret message, he/she still cannot extract it without the correct user key.

For secret authentication, we have proposed a method for authentication of secret messages for fidelity verification in PDF files. We use another data hiding technique for modifying the generation numbers of an indirect object in the PDF file. By this

method, we can authenticate the secret message for fidelity verification. As long as the secret message has been tampered with by an illicit user, the authentication will fail. We use two bytes to embed the authentication signal in this study; however, if the secret message is long, we can generate more bytes of the authentication signals for secret authentication.

For secret sharing, we have proposed a method via PDF documents by the data hiding technique mentioned before. In this method, the secret which we want to share is a PDF document. We can select cover PDFs and user keys randomly. After applying the proposed secret sharing algorithm, we can get shares and distribute them to a group of participants of the same number, each participant with a share. If we want to recover the secret PDF document, we can collect the shares from each participant with the correct user key and apply the proposed recovery algorithm to get it. Even if an illicit user steals all the shares, he/she still cannot recover the secret PDF document without the correct user keys from other participants. The method offers a way to protect the PDF documents which are not owned by single persons but a group of people in cooperation.

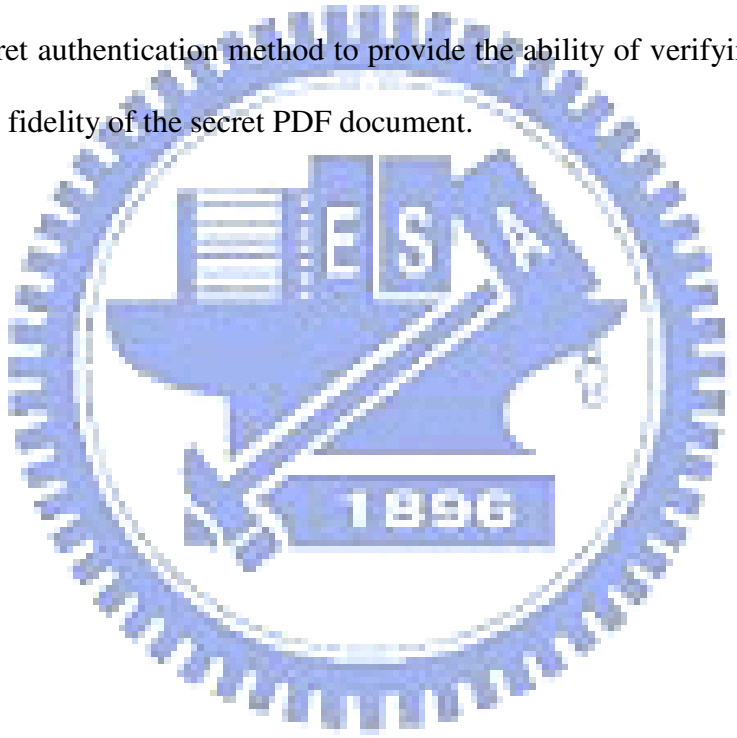
6.2 Suggestions for Future Works

Several suggestions for future research works are listed as follows.

1. The proposed data hiding technique of space coding may be improved by inserting more white-space characters to raise the capacity of the embedded data; or by randomizing the white-space characters which we use for embedding data to strengthen the security of the hidden data.
2. The proposed data hiding technique of inserting invisible texts may be improved by modifying more parameters of text matrices for embedding

secret messages to reduce the redundancy of the embedding data.

3. The proposed secret transmission method may be improved by hiding secret messages in other types of objects to strengthen the security of the hidden data.
4. The proposed secret authentication method may be improved by generating more authentication signals of the secret message to provide the ability of verifying the integrity.
5. The proposed secret sharing method may be integrated with the proposed secret authentication method to provide the ability of verifying the integrity and fidelity of the secret PDF document.



References

[1] Adobe Systems Incorporated, *PDF Reference*, Sixth Edition, Addison-Wesley, California, USA, Nov. 2006.

[2] H. K. Pan, Y. Y. Chen, and Y. C. Tseng, "A Secure Data Hiding Scheme for Two-Color Images," *IEEE ISCC 2000*, Antibes-Juan les Pins, France, pp. 750-755, July 1999.

[3] J. J. Chae and B. S. Manjunath, "Data Hiding in Video," *Proceedings of IEEE International Conference of Image Processing*, Kobe, Japan, vol. 1, pp. 311-315, Dec 1999.

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

[5] Jeanne Chen , Tung-Shon Chen and Meng-Wen Cheng, "A New Data Hiding Method in Binary Image," *Multimedia Software Engineering* ,Proceedings. Fifth International Symposium, 2003, pp. 88 – 93.

[6] T. Y. Liu and W. H. 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, pp. 24-30, March, 2007.

[7] C. C. Wang and W. H. Tsai, "Creation of Tile-overlapping mosaic images for information hiding," *Proceedings of 2007 National Computer Symposium,* Taichung, Taiwan, R. O. C., pp. 119-126, December, 2007.

[8] G. L. Huang and W. H. Tsai, "Optimal Data Hiding in H.264/AVC Videos for Covert Communication," *Proceedings of 2008 Conference on Computer Vision, Graphics and Image Processing*, Yilan, Taiwan, R. O. C., Aug. 24-26, 2008.

[9] I. S. Lee and W. H. Tsai, "Data hiding in grayscale images by dynamic programming based on a human visual model," *Pattern Recognition*, Vol. 42, No. 7, pp. 1604-1611. July, 2009.

[10] S. Zhong and T. Chen, "Information steganography algorithm based on PDF documents," *Computer Engineering*, vol. 32, No. 3, pp. 161-163, Feb. 2006.

[11] 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, No. 1, pp. 17-26, January, 2007.

[12] X. Liu et al., "A steganographic algorithm for hiding data in PDF files based on equivalent transformation," *Proceedings of 2008 International Symposiums on Information Processing (ISIP)*, pp. 417-421, Moscow, Russia, May 23-25, 2008.

[13] J. T. Wang and W. H. Tsai, "Data hiding in PDF files and applications by imperceptible modifications of PDF object parameters," *Proceedings of 2008 Conference on Computer Vision, Graphics and Image Processing*, Yilan, Taiwan, R. O. C., August, 24-26, 2008.

[14] G. R. Blakley, "Safeguarding Cryptographic Keys," *Proceedings of AFIPS National Computer Conference,* Vol. 48, pp. 313-317, New York, U.S.A., June 4-7, 1979.

[15] A. Shamir, "How to share a secret," *Communications of Association for Computing Machinery* , Vol. 22, No. 11, pp. 612- 613, Nov., 1979.

[16] C. C. Lin and W. H. Tsai, "Secret multimedia information sharing with data hiding capacity by simple logic operations," *Proceedings of 5th World Multiconference on Systemics, Cybernetics, and Informatics, Vol. I: Information Systems Development*, pp. 50-55, Orlando, Florida, U. S. A., 2001.

[17] K. L. Huang, "Secret Sharing with Steganographic Effects for HTML Documents," *Proceedings of 2004 Conference on Computer Vision, Graphics*

and Image Processing, Hualien, Taiwan, R. O. C., Aug. 15-17, 2004.

[18] K. L. Huang, "A study on information sharing of text-type documents with steganography and authentication capabilities," *MS Thesis,* Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, R. O. C., June 2004.

[19] G. L. Huang, "A study on data hiding in H.264-AVC videos and its applications," *MS Thesis,* Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, R. O. C., June 2008.