International Journal of Innovative Computing, Information and Control
Volume **9**, Number **1**, January **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 1–16

A COPYRIGHT PROTECTION SCHEME BASED ON PDF

Hsiu-Feng Lin[1], Li-Wei Lu[1], Chiou-Yueh Gun[2] and Chih-Ying Chen[3]

[1]Department of Information Engineering and Computer Science
[3]Department of Communications Engineering
Feng-Chia University
No. 100, Wenhwa Rd., Seatwen, Taichung 40724, Taiwan
{ hflin; chihchen }@fcu.edu.tw; vacuus@livemail.tw

[2]Department of Mechanical Engineering
Nan-Kai University of Technology
No. 568, Zhongzheng Rd., Caotun Township, Nantou County 54243, Taiwan
moon384@nkut.edu.tw

Received July 2011; revised April 2012

Abstract. *The information hiding method in this study was based on PDF files of iso-8859-1 encoding. The hiding method that we discovered can hide information within PDF files, and the hidden information will not be detected of any irregularity by users when reading the PDF files. We developed an encryption technique which combines the information hiding technology of PDF documents and quadratic residue as basis, and applies to copyright protection and digital learning. The copyright owner can use a personal secret key to randomly generate a sequence of characters in PDF documents (or pixels in digital images) to operate with the pixels in the watermark to create verification ciphertexts. The watermark, digital content in PDF format and ciphertext verification are then stored at the copyright registration center. Moreover, by using our scheme, once piracies with the same serial numbers as genuine products are found, their sources can be traced. Furthermore, this encryption technique can also apply to digital learning. The learning questions and answers are first categorized according to their level, in which the characters of the questions and answers are then created with hidden ciphertext. This can prevent the learner from directly seeing the answer before thinking.*

Keywords: Portable Document Format (PDF), Information hiding, Quadratic residue, Factorization problems

1. **Introduction.** Briefly speaking, information hiding technology is to hide secret data in a file, of which the file would appear normal and show no trace of irregularity to most people. Both sides involved with the hidden information would need to first communicate with each other in how to retrieve the secret data hidden within the file. The main purpose for hiding information is to prevent a third person (party) from knowing about the passing data. Therefore, the hiding of information is a kind of communicating to prevent data from being damaged, modified, or the retrieval of secret data from a third person (party).

Digital watermarking is a technology which revises the pixel values of digital images to hide important information. In general, the revised digital image usually has little difference with the original image, which the human eye basically would find difficult to identify the difference between the original image and the revised one. The watermark in the original image can only be retrieved if the owner uses the secret key.

Various types of documental formats are used among our daily lives, such as TXT, DOC and PDF. Because PDF files are commonly used by most users and well publicized by Adobe [1] vendors, we therefore chose to use PDF files for information hiding study.

Our information hiding technology is applicable for PDF files of iso-8859-1 [2] encoding. It is also combined with a copyright protection technology based on quadratic residue [3]. Because the copyright owner is the only one who knows the secret key, a person would encounter a factorization problem if the data were intended to fabricate.

In 2007, Castiglione et al. [4] used the reserved header space of documents in Microsoft Office Word to hide information. The hided information showed to have no effect to the opening and reading of the document. In the same year, Zhong et al. [5], used a technique according to the preset distance between each word in PDF document management for information hiding, which also showed no influence to the reading quality of the document. In 2010, Lee and Tsai [6] discovered that the hexadecimal text encoding of "20" and "A0" are displayed as blank characters in PDF documents based on iso-8859-1 character encoding [2]. According to iso-8859-1 [2] encoding, the preset blank character encoding in PDF is "20" and therefore the first hiding method is to respectively use "20" and "A0" as the binary digits of "0" and "1", in which then the original blank character code is replaced. After the replacement, the display of blank characters in PDF remains the same. The second hiding method uses the following steps: (i) set the font size of "A0" as 0, (ii) encode each character of the hidden information into the Huffman code, in which each codeword is expressed with the string series of "A0", (iii) the space between words in PDF is expressed by "20", and the codeword is inserted inside the text between the characters in words. The above two methods have the advantage of displaying identical results in PDF reader before and after the hiding, and showing no influence when reading the PDF file. However, the method by Lee and Tsai [6] would be more difficult to accomplish if the PDF document contains no text but simply images.

2. Mathematical Background and Introduction to PDF Syntax.

2.1. Quadratic residue.

Quadratic residue [3] is frequently used in cryptography. Suppose n is a positive integer, a is called a quadratic residue modulo n if $\gcd(a, n) = 1$, and $x^2 \equiv a \bmod n$ is solvable. Otherwise, a is called a quadratic non-residue modulo n. The set of quadratic residues of Z_n is denoted by QR_n and the set of quadratic non-residue by QNR_n.

Definition 2.1. *Let p be an odd prime number and let $a \in Z$. The Legendre symbol denoted as, $\left(\frac{a}{p}\right)$, is defined by* $\left(\frac{a}{p}\right) = \begin{cases} 1, & \textit{if } a \textit{ is called a quadratic residue modulo } p \\ -1, & \textit{if } a \textit{ is called a quadratic non-residue modulo } p \\ 0, & \textit{if } a \textit{ divides } p \end{cases}$

By Fermat theorem, we have the following:

Proposition 2.1. *Suppose that p is an odd prime number and a, b are positive integers, then the following statements are true.*

(1) $a \in QR_p$ iff $a^{\frac{p-1}{2}} \equiv 1 \bmod p$, and $a \in QNR_p$ iff $a^{\frac{p-1}{2}} \equiv -1 \bmod p$.

(2) $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$.

(3) $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$ iff $a \equiv b \bmod p$.

By (2) of Proposition 2.1, we have the conclusion that the product of two quadratic residues is also a quadratic residue and the product of two quadratic non-residues is also a quadratic residue. Yet the product of a quadratic residue and a quadratic non-residue is a quadratic non-residue.

Proposition 2.2. *Let p and q be two large prime numbers, and $n = p \times q$, define the set $Z_n^* = \{a \in Z : 0 < a < n$ and $(a, n) = 1\}$, and then Z_n^* can be divided into four* *equivalence classes as follows:*

$$Z_{(1,1)} = \left\{ a \in Z_n^* \,\middle|\, \left(\frac{a}{p}\right) = 1, \left(\frac{a}{q}\right) = 1 \right\}$$
$$Z_{(1,-1)} = \left\{ a \in Z_n^* \,\middle|\, \left(\frac{a}{p}\right) = 1, \left(\frac{a}{q}\right) = -1 \right\}$$
$$Z_{(-1,1)} = \left\{ a \in Z_n^* \,\middle|\, \left(\frac{a}{p}\right) = -1, \left(\frac{a}{q}\right) = 1 \right\}$$
$$Z_{(-1,-1)} = \left\{ a \in Z_n^* \,\middle|\, \left(\frac{a}{p}\right) = -1, \left(\frac{a}{q}\right) = -1 \right\}.$$

Therefore, $a \in QR_n$ iff $a \in Z_{(1,1)}$ and $a \in QNR_n$ iff $a \in Z_{(1,-1)} \cup Z_{(-1,1)} \cup Z_{(-1,-1)}$. Let c_1, c_2, c_3, c_4 be defined as $c_1 \in Z_{(1,1)}$, $c_2 \in Z_{(1,-1)}$, $c_3 \in Z_{(-1,1)}$, $c_4 \in Z_{(-1,-1)}$. Then for any $a \in Z_n^*$, we can choose a suitable number c_i, $i \in \{1, 2, 3, 4\}$ such that $c_i a \in Z_{(1,1)}$. For example, if we choose p and q such that $p \equiv 3 \pmod 8$ and $q \equiv 7 \pmod 8$, we may set $c_1 = 1$, $c_2 = -2$, $c_3 = 2$, $c_4 = -1$. The following are properties of quadratic residues mod n.

(1) For $a \in Z_n^*$, $a \in QR_n$ iff $a^{-1} \in QR_n$.

(2) $a \in QR_n$ iff $a \in QR_p \cap QR_q$.

(3) If a is the quadratic residue mod n, where $n = p \times q$, and p, q are known in advance, then the four roots of $z^2 \equiv a \bmod n$ can be worked out in polynomial time.

Theorem 2.1. *Let $n = p \times q$ where $p \equiv q \equiv 3 \pmod 4$. If $a \in QR_n$ and $(a, n) = 1$, then each of the four roots of $x^2 \equiv a \bmod n$ fits in $Z_{(1,1)}$, $Z_{(1,-1)}$, $Z_{(-1,1)}$, $Z_{(-1,-1)}$, respectively. Furthermore, $x^{2^k} \equiv a \bmod n$ has a solution for every integer $k \geq 1$.*

Note: If $p \equiv 3 \pmod 4$ is a prime and $a \in QR_p$, then a has a square root x $(= \sqrt{a}) = a^{\frac{p+1}{4}} \pmod p$, (since $a^{p+1} \equiv a^2 \pmod p$ and $-1 \in QNR_p$).

2.2. **Introduction to PDF syntax.**

We only introduce a small part of the PDF syntax, in which only the syntaxes which controls text and images in PDF are introduced. For details of the complete PDF specification, please see PDF Reference, Sixth Edition, version 1.7 [7].

2.2.1. *PDF text syntax.* An example of a PDF syntax is given in Table 1. In the example, "BT" represents Begin Text, and "ET" represents End the Text. The datum point $(0, 0)$ for the displaying coordinates of the first row between "BT" and "ET" is located at the

TABLE 1. Example of PDF text syntax

syntax
BT
/F13 12 Tf
288 720 Td
(ABC) Tj
−5 −18 Td
(DEF)Tj
/F13 18 Tf
0 −18 Td
(IJK) Tj
ET

bottom left corner refers to the original coordinate $(0, 0)$ of the whole PDF document page. The displaying coordinates for the text of the second row and so on is based on shifting the datum point of the previous text row. Syntax "/F13 12 Tf" is used for setting the text style and font size, in which "F13" sets the text style and "12" sets the font size. This text setting syntax will apply to the current text and till the next text setting syntax; "288 720 Td" refers to the displaying coordinates, and "(ABC) Tj" is to display the text "ABC". As a result, the document "ABC" will display at a coordinate position $(288, 720)$, with text style of "F13" and font size "12".

Syntax "$-5\ -18$ Td" refers to displaying "DEF" at coordinate $(-5, -18)$. Because "DEF" is not the first row, the datum point for the displaying coordinate is based on the first character of the previous row text referring to the new original coordinate $(0, 0)$, as shown in the dotted arrow of Figure 1. As a result, the displaying position of "DEF" is based on the new datum point of the previous row, shifting 5 units left and 18 units downwards. Therefore, "DEF" will display at a position beneath "ABC" to the left with text style "F13" and font size "12", i.e., The text between "/F13 12 Tf" and "/F13 18 Tf" is based on "/F13 12 Tf" setting. Furthermore, any text after syntax "/F13 18 Tf" will follow the setting, hence the text style for "IJK" is "F13" with font size "18". The final results are shown in Figure 1. If syntax "/F13 18 Tf" did not exist, then "IJK" would have applied to the text setting syntax of "/F13 12 Tf".

2.2.2. *PDF image syntax.* The syntax "Im1" in Table 2 is the name of the image, "Do" refers to draw the image of "Im1", and "132 0 0 132 45 140 cm" indicates the width and height setting for the image at 132 units and to display at position $(45, 140)$. Furthermore, "q" commands the image status to become stacked prior to drawing, and "Q" commands the image to restore its compressed state after the drawing is completed.

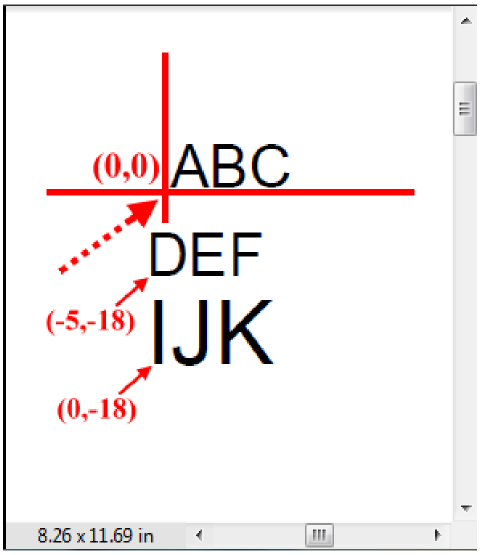


FIGURE 1. Display results of the PDF text syntax example in Table 1

TABLE 2. PDF image syntax example

syntax
q 132 0 0 132 45 140 cm /Im1 Do Q

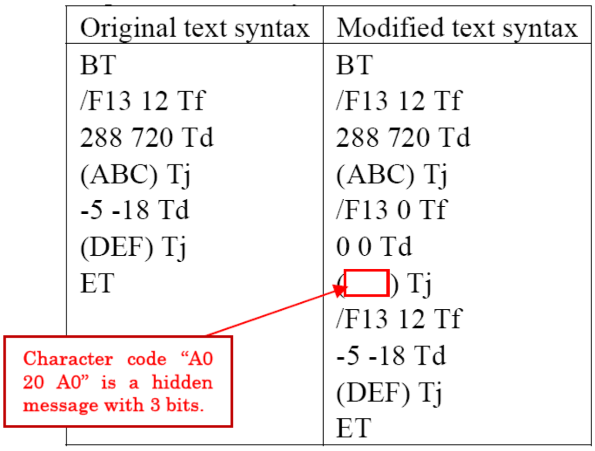
3. Research Method.

3.1. PDF information hiding method – for texts. Lee and Tsai [6] indicated that the hexadecimal character codes of “20” and “A0” in a PDF document are displayed as blank characters according to iso-8859-1 [2] encoding, and can be used to represent the binary digits of “0” and “1”. As a result, the method can only be used for PDF files under the encoding format of iso-8859-1 [2]. We assume that the secret data for hiding was of binary digits “101”. Therefore, the data can be expressed in hexadecimal character code “A0 20 A0”.

From the comparison of text syntax modification shown in Table 3, we can notice the three blank characters in the modified syntax of “( ) Tj”. Based on iso-8859-1 [2] encoding, the blank characters represent the hexadecimal character code of “A0 20 A0”, and “0 0 Td” represent the displaying coordinate $(0, 0)$ for the three blank characters. Since the blank characters are not on the first row, the datum point of the displaying coordinates is based from the previous row. As a result, the three blank characters will stack up on “ABC” under PDF display. The purpose for syntax “/F13 0 Tf” is to have the three blank characters apply with font style “F13” and font size of “0”. The key for the three blank characters lies in the font size setting as “0”. The three blank characters will completely disappear. The text “DEF” originally applied a text setting of “/F13 12 Tf”; however as we inserted “/F13 0 Tf”, “DEF” shall apply the “/F13 0 Tf” setting. We therefore need to additionally insert “/F13 12 Tf” above “5 -18 Td” to maintain the original text setting for “DEF”. As a result, the displaying appearance from the original text syntax and modified text syntax in PDF Reader will have completely same results. The three blank characters by adding “( ) Tj”, with character code “A0 20 A0”, have become a hiding data.

3.2. PDF information hiding method – for images. If we intend to hide a secret binary digit data “101” within a PDF image, an additional text syntax is added to the original image syntax, as in Table 4. We therefore need to insert a syntax block which begins with “BT” and ends with “ET”. The three blank characters in the modified syntax “( ) Tj” represent the hexadecimal character code of “A0 20 A0” according to iso-8859-1 [2] encoding, and the displaying position of the blank characters are the same to the displaying position setting for “Im1” is at $(45, 140)$. As a result, the three blank characters will stack with “Im1”, and the blank characters apply to the text setting of “/F13 0 Tf”.

Table 3. Comparison of text syntax before and after modification



Original text syntax	Modified text syntax
BT	BT
/F13 12 Tf	/F13 12 Tf
288 720 Td	288 720 Td
(ABC) Tj	(ABC) Tj
-5 -18 Td	/F13 0 Tf
(DEF) Tj	0 0 Td
ET	( ) Tj
	/F13 12 Tf
	-5 -18 Td
	(DEF) Tj
	ET

Character code “A0 20 A0” is a hidden message with 3 bits.

Table 4. Comparison of image syntax before and after modification

Original image syntax	Modified syntax
q	q
132 0 0 132 45 140 cm	132 0 0 132 45 140 cm
/Im1 Do	/Im1 Do
Q	Q
	BT
	/F13 0 Tf
	45 140 Td
	( ) Tj
	ET

Character code "A0 20 A0" is a hidden message with 3 bits.

3.3. **Retrieval of hidden information from PDF text documents.** The first step was to search if syntax "0 0 Td" appears in the text syntax. The next step was to check if the next syntax ends with "Tj". If the syntax meets with both conditions, the text character code is the hidden information. After the data is retrieved, the search will continue to find the next "0 0 Td", and the process goes on until the whole text syntax was searched. The retrieved character codes are then combined together according to the sequence, in which "20" and "A0" respectively represent the binary digits of "0" and "1". The assembled character code is then converted back to the binary digit data message which we want.

3.4. **Retrieval of hidden information from PDF image documents.** If the hidden information is concealed in image "Im1", as in Table 4, we need to search the additional syntax block in order to retrieve the hidden information. The search method is to find the additional syntax block, which begins as a text setting syntax at the first row with font size set to 0 between "BT" and "ET", and the second row is to continue as a text displaying coordinate syntax, the displaying coordinate is required to be as same as the displaying coordinate of "Im1". When the syntax block is found, the text character code in the text display syntax is our hidden information. We retrieve all the character codes from the syntax blocks and combine them together according to the sequence, in which "20" and "A0" respectively represent the binary digits of "0" and "1". The assembled character code is then converted back to the binary digit data message which we want.

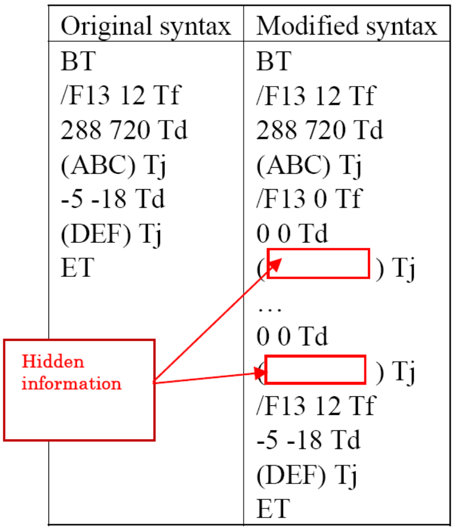
3.5. **Application of PDF information hiding method applied to copyright protection – for texts.** A PDF file in accordance to iso-8859-1 [2] encoding includes m characters and watermark $W = w_1w_2w_3\dots w_{2t-1}w_{2t}$, $w_i \in \{0,1\}$, and $n = p \times q$, where p, q are strong primes which satisfy $p \equiv 7 \pmod 8$ and $q \equiv 3 \pmod 8$; p, q are the secret keys which should be kept from (unknown to) others.

Algorithm I

```
Step 1: An integer from $\overline{x_0} \in Z_n^*$ is randomly selected which there exists a unique $b \in \{1,-1,2,-2\}$, so that $x_0 = b\overline{x_0}$, with $x_0 \in QR_n$. The value $x_0$ is an initial hidden value which should be kept from (unknown to) others.
Step 2: Run $i = 1$ to $t$
    (1) Find the solution of $x_i^2 \equiv x_{i-1} \pmod n$, satisfying $x_i \in QR_n$.
    (2) Calculate $u_i = x_i \bmod m$.
    (3) Retrieve two of the most significant bits of the $u_i$ character among the $m$ characters, assigned as $k_{2i-1}$ and $k_{2i}$.
    (4) Calculate $c_{2i-1} = k_{2i-1} \oplus w_{2i-1}$ and $c_{2i} = k_{2i} \oplus w_{2i}$.
```

Step 3: Calculate $x_{t+1}^2 \equiv x_t \pmod{n}$, satisfying $x_t \in QR_n$.
Step 4: We can then obtain $S = (x, c)$, where $x = x_{t+1}$, $c = c_1 c_2 \ldots c_{2t}$.
Step 5: We would then want to hide the entire S into the PDF file. Based on iso-8859-1 [2] encoding, we convert the binary digit "1" into the hexadecimal character code "A0", and binary digit "0" into the hexadecimal character code "20". After S is converted into binary digits, the text setting syntax insert is inserted with syntax "/F13 0 Tf", using every 8 bits as a unit and syntax "( ) Tj" as a set. An example is shown in Table 5.

TABLE 5. Example of hiding information into a text syntax


Hidden
information

Original syntax	Modified syntax
BT	BT
/F13 12 Tf	/F13 12 Tf
288 720 Td	288 720 Td
(ABC) Tj	(ABC) Tj
-5 -18 Td	/F13 0 Tf
(DEF) Tj	0 0 Td
ET	( ) Tj
	…
	0 0 Td
	( ) Tj
	/F13 12 Tf
	-5 -18 Td
	(DEF) Tj
	ET

Step 6: The PDF file, watermark $W = w_1 w_2 w_3 \ldots w_{2t-1} w_{2t}$, and $S = (x, c)$ are sent to the copyright registration center for registration.

Remark 3.1. *Large amounts of data can be hidden by setting the font size of the hidden data to "zero". This will not affect the displaying results in PDF Reader.*

Remark 3.2. *The watermark ($W = w_1 w_2 \ldots w_{2t}$) length is set to $2t$, having m characters in the PDF text file. From Step 2 of Algorithm I, it can be seen that the characters in the PDF file are capable of performing "$\oplus$" operations with the pixels in the watermark repeatedly. Because the watermark length, $2t$, is irrelevant to the size of m, the algorithm is operable even when $2t$ is greater than m.*

3.6. **Application of PDF information hiding method applied to copyright protection – for images.** A PDF file in accordance to iso-8859-1 [2] encoding includes an image. The image consists of m pixels and watermark $W = w_1 w_2 w_3 \ldots w_{2t-1} w_{2t}$, $w_i \in \{0, 1\}$, and $n = p \times q$, where p, q are strong primes which satisfy $p \equiv 7 \pmod{8}$ and $q \equiv 3 \pmod{8}$; p, q are the secret keys which should be kept from (unknown to) others.

Algorithm II

Step 1: An integer from $\overline{x_0} \in Z_n^*$ is randomly selected which there exists a unique $b \in \{1, -1, 2, -2\}$ so that $x_0 = b\overline{x_0}$, with $x_0 \in QR_n$. The value x_0 is an initial hidden value which should be kept from (unknown to) others.
Step 2: Run $i = 1$ to t
(1) Find the solution for $x_i^2 \equiv x_{i-1} \pmod{n}$ which also satisfies $x_i \in QR_n$.
(2) Calculate $u_i = x_i \bmod m$.
(3) Retrieve the two fixed most significant bits of the u_i character among the m characters, assigned as k_{2i-1} and k_{2i}. The reason for retrieving two fixed most

significant bits is because only the least significant bits are affected if the image is slightly altered. The watermark would therefore still be retrievable during verification. If the image was excessively altered, the high bytes are affected, and people can easily detect that the image has been altered.

(4) Calculate $c_{2i-1} = k_{2i-1} \oplus w_{2i-1}$ and $c_{2i} = k_{2i} \oplus w_{2i}$

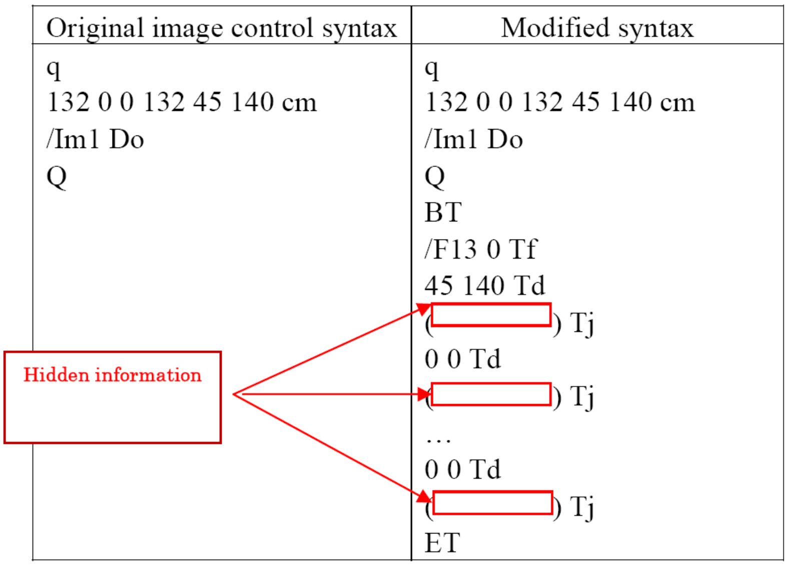
Step 3: Calculate $x_{t+1}^2 \equiv x_t \pmod n$, satisfying $x_t \in QR_n$.

Step 4: We can then obtain $S = (x, c)$, where $x = x_{t+1}$, $c = c_1 c_2 \ldots c_{2t}$

Step 5: We would then want to hide the entire S into the PDF file. After S is converted into binary digits, the text setting syntax insert is inserted with syntax "/F13 0 Tf", using every 8 bits as a unit and syntax "( ) Tj" as a set. The modification of the syntax is given in Table 6. The displaying coordinate for the first row text is $(45, 140)$. Since the displaying coordinate of the image in the PDF file is also $(45, 140)$, the first row blank will cover onto the image, and the displaying coordinate for the second row text is $(0, 0)$. The second row will therefore cover onto the first row, and the third row will also stack onto the original position and so on. As a result, the whole set of blank characters will stack onto the image, where all of the blank characters apply the setting of "/F13 0 Tf".

Step 6: The PDF file, watermark $W = w_1 w_2 w_3 \ldots w_{2t-1} w_{2t}$, and $S = (x, c)$ are sent to the copyright registration center for registration .

Table 6. Example of hiding information into the image syntax



Original image control syntax	Modified syntax
q 132 0 0 132 45 140 cm /Im1 Do Q	q 132 0 0 132 45 140 cm /Im1 Do Q BT /F13 0 Tf 45 140 Td ( ) Tj 0 0 Td ( ) Tj … 0 0 Td ( ) Tj ET

Hidden information

3.7. **PDF copyright verification – for texts.** We discuss the situation of a PDF document purchased from the copyright owner which follows iso-8859-1 [2] encoding with embedded verification information, $S = (x, c)$, $c = c_1 c_2 \ldots c_{2t}$. The data include m characters with a public key $n = p \times q$.

Algorithm III

Step 1: Run $i = t$ to 1.

(1) Calculate $x_i = x_{i+1}^2 \pmod n$.

(2) Calculate $u_i = x_i \pmod m$.

(3) Retrieve two of the most significant bits of the u_i character among the m characters, assigned as k_{2i-1} and k_{2i}.

(4) Calculate $w_{2i-1} = k_{2i-1} \oplus c_{2i-1}$ and $w_{2i} = k_{2i} \oplus c_{2i}$.

```
Step 2: The watermark is then obtained, $W = w_1 w_2 w_3 ... w_{2t-1} w_{2t}$.
Step 3: Compare the obtained watermark with the original watermark registered at the
        copyright registration center to see if they are the same.
```

3.8. **PDF copyright verification – for images.** We discuss the situation of a PDF document purchased from the copyright owner which follows iso-8859-1 [2] encoding which contains an image. The image includes m pixels with a public key $n = p \times q$; $S = (x, c)$, $x = x_{t+1}$, $c = c_1 c_2 \dots c_{2t}$.

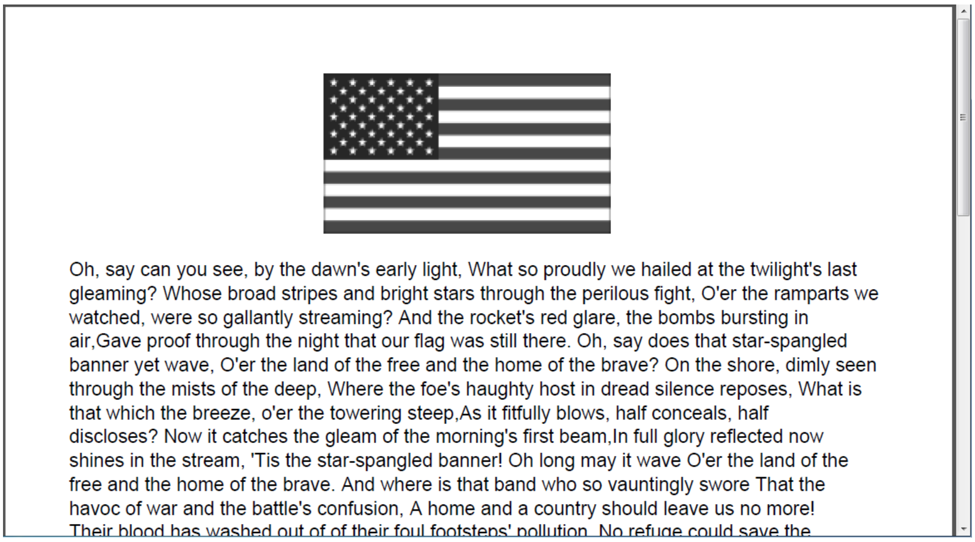
Algorithm IV

```
Step 1: Run $i = t$ to 1.
        (1) Calculate $x_i = x_{i+1}^2 \pmod{n}$.
        (2) Calculate $u_i = x_i \pmod{m}$.
        (3) Retrieve two of the most significant bits of the $u_i$ pixel among the $m$ pixels,
            assigned as $k_{2i-1}$ and $k_{2i}$.
        (4) Calculate $w_{2i-1} = k_{2i-1} \oplus c_{2i-1}$ and $w_{2i} = k_{2i} \oplus c_{2i}$.
Step 2: The watermark is then obtained, $W = w_1 w_2 w_3 \dots w_{2t-1} w_{2t}$.
Step 3: Compare the obtained watermark with the original watermark registered at the
        copyright registration center to see if they are the same.
```

Remark 3.3. *A watermark can be created by just having the verification ciphertext and digital image. The verification ciphertext and digital image need to be just expressed in PDF format.*

4. **Experiment Results and Applications.** We will conduct some experiments in the following paragraphs according to the research method previously described. We prepared a PDF document based on iso-8859-1 [2] encoding. The file includes a gray scale American flag image. The size of the flag is 180 × 100, and the text is the American anthem, as shown in Figure 2.

The following description is the testing of the watermark. The size of the watermark is 100 × 50, as in Figure 3. The program used in this study was written in C#, and iTextSharp was used to process API for the PDF document [8].



Oh, say can you see, by the dawn's early light, What so proudly we hailed at the twilight's last gleaming? Whose broad stripes and bright stars through the perilous fight, O'er the ramparts we watched, were so gallantly streaming? And the rocket's red glare, the bombs bursting in air,Gave proof through the night that our flag was still there. Oh, say does that star-spangled banner yet wave, O'er the land of the free and the home of the brave? On the shore, dimly seen through the mists of the deep, Where the foe's haughty host in dread silence reposes, What is that which the breeze, o'er the towering steep,As it fitfully blows, half conceals, half discloses? Now it catches the gleam of the morning's first beam,In full glory reflected now shines in the stream, 'Tis the star-spangled banner! Oh long may it wave O'er the land of the free and the home of the brave. And where is that band who so vauntingly swore That the havoc of war and the battle's confusion, A home and a country should leave us no more! Their blood has washed out of of their foul footsteps' pollution. No refuge could save the

Figure 2. PDF document example

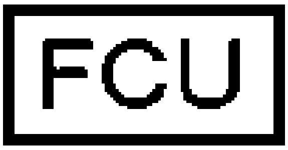
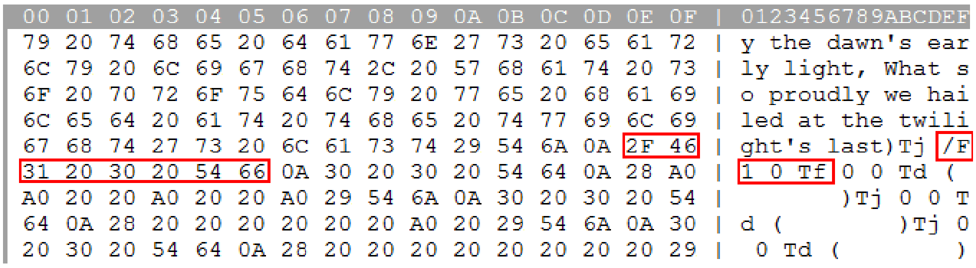
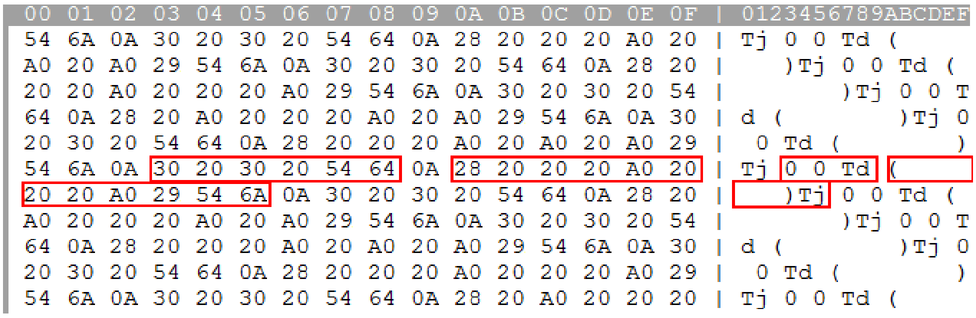


Figure 3. Watermark



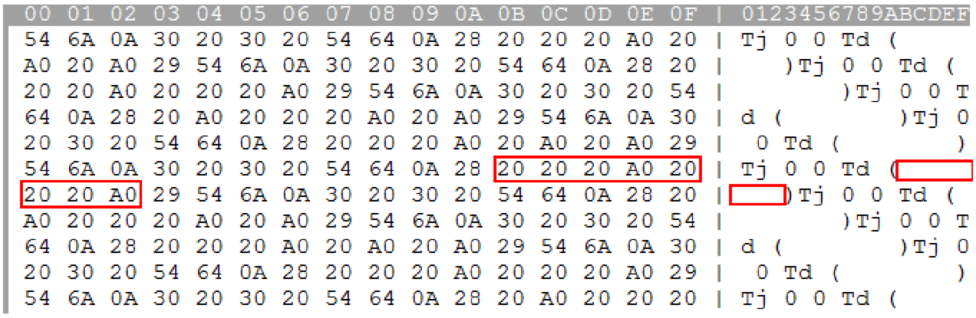
```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F | 0123456789ABCDEF
79 20 74 68 65 20 64 61 77 6E 27 73 20 65 61 72 | y the dawn's ear
6C 79 20 6C 69 67 68 74 2C 20 57 68 61 74 20 73 | ly light, What s
6F 20 70 72 6F 75 64 6C 79 20 77 65 20 68 61 69 | o proudly we hai
6C 65 64 20 61 74 20 74 68 65 20 74 77 69 6C 69 | led at the twili
67 68 74 27 73 20 6C 61 73 74 29 54 6A 0A 2F 46 | ght's last)Tj /F
31 20 30 20 54 66 0A 30 20 30 20 54 64 0A 28 A0 | 1 0 Tf 0 0 Td (
A0 20 20 A0 20 20 A0 29 54 6A 0A 30 20 30 20 54 |        )Tj 0 0 T
64 0A 28 20 20 20 20 20 20 A0 20 29 54 6A 0A 30 | d (         )Tj 0
20 30 20 54 64 0A 28 20 20 20 20 20 20 20 20 29 |  0 Td (        )
```

Figure 4. PDF syntax observation – text setting syntax



```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F | 0123456789ABCDEF
54 6A 0A 30 20 30 20 54 64 0A 28 20 20 20 A0 20 | Tj 0 0 Td (
A0 20 A0 29 54 6A 0A 30 20 30 20 54 64 0A 28 20 |    )Tj 0 0 Td (
20 20 A0 20 20 20 A0 29 54 6A 0A 30 20 30 20 54 |        )Tj 0 0 T
64 0A 28 20 A0 20 20 20 A0 20 A0 29 54 6A 0A 30 | d (         )Tj 0
20 30 20 54 64 0A 28 20 20 20 A0 20 A0 20 A0 29 |  0 Td (        )
54 6A 0A 30 20 30 20 54 64 0A 28 20 20 20 A0 20 | Tj 0 0 Td (
20 20 A0 29 54 6A 0A 30 20 30 20 54 64 0A 28 20 |    )Tj 0 0 Td (
A0 20 20 20 A0 20 A0 29 54 6A 0A 30 20 30 20 54 |        )Tj 0 0 T
64 0A 28 20 20 20 A0 20 A0 20 A0 29 54 6A 0A 30 | d (         )Tj 0
20 30 20 54 64 0A 28 20 20 20 A0 20 20 20 A0 29 |  0 Td (        )
54 6A 0A 30 20 30 20 54 64 0A 28 20 A0 20 20 20 | Tj 0 0 Td (
```

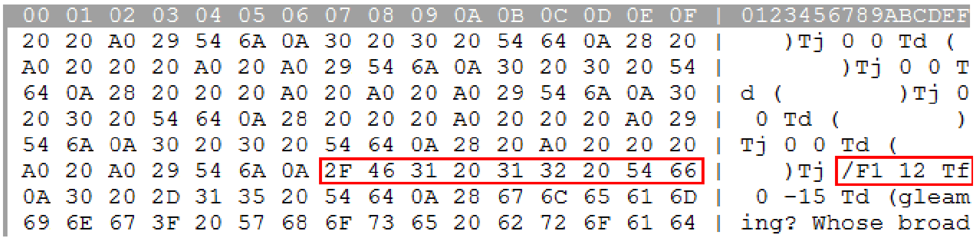
Figure 5. PDF syntax observation – block syntax



```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F | 0123456789ABCDEF
54 6A 0A 30 20 30 20 54 64 0A 28 20 20 20 A0 20 | Tj 0 0 Td (
A0 20 A0 29 54 6A 0A 30 20 30 20 54 64 0A 28 20 |    )Tj 0 0 Td (
20 20 A0 20 20 20 A0 29 54 6A 0A 30 20 30 20 54 |        )Tj 0 0 T
64 0A 28 20 A0 20 20 20 A0 20 A0 29 54 6A 0A 30 | d (         )Tj 0
20 30 20 54 64 0A 28 20 20 20 A0 20 A0 20 A0 29 |  0 Td (        )
54 6A 0A 30 20 30 20 54 64 0A 28 20 20 20 A0 20 | Tj 0 0 Td (
20 20 A0 29 54 6A 0A 30 20 30 20 54 64 0A 28 20 |    )Tj 0 0 Td (
A0 20 20 20 A0 20 A0 29 54 6A 0A 30 20 30 20 54 |        )Tj 0 0 T
64 0A 28 20 20 20 A0 20 A0 20 A0 29 54 6A 0A 30 | d (         )Tj 0
20 30 20 54 64 0A 28 20 20 20 A0 20 20 20 A0 29 |  0 Td (        )
54 6A 0A 30 20 30 20 54 64 0A 28 20 A0 20 20 20 | Tj 0 0 Td (
```

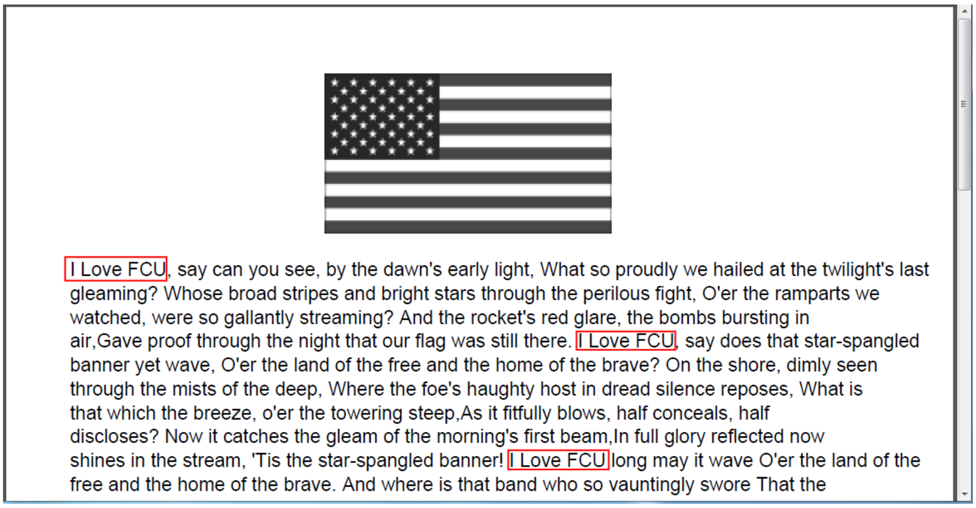
Figure 6. PDF syntax observation – information hiding

4.1. **PDF information hiding method applied to copyright protection – Experiment 1.** We conduct an experiment according to the method described in Section 3.5. The text in the PDF document example is displayed with "F1" font and font size "12". We want to hide S between the first and second row text. The procedure is shown in Figure 4, as shown in the figure, we first need to insert syntax "/F1 0 Tf" after the first row text syntax. The framed area shown in Figure 5 indicates the message syntax block. The 8 space blanks in "( ) Tj" consisted by the hexadecimal character codes of "20" and "A0" in accordance with iso-8859-1 [2] encoding, as shown in Figure 6. Finally, we then add syntax "/F1 12 Tf", to maintain the original text setting after the second row, as in Figure 7.



```
 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F | 0123456789ABCDEF
 20 20 A0 29 54 6A 0A 30 20 30 20 54 64 0A 28 20 |    )Tj 0 0 Td (
 A0 20 20 20 A0 20 A0 29 54 6A 0A 30 20 30 20 54 |        )Tj 0 0 T
 64 0A 28 20 20 20 A0 20 A0 20 A0 29 54 6A 0A 30 | d (        )Tj 0
 20 30 20 54 64 0A 28 20 20 20 A0 20 20 20 A0 29 |  0 Td (         )
 54 6A 0A 30 20 30 20 54 64 0A 28 20 A0 20 20 20 | Tj 0 0 Td (
 A0 20 A0 29 54 6A 0A 2F 46 31 20 31 32 20 54 66 |    )Tj /F1 12 Tf
 0A 30 20 2D 31 35 20 54 64 0A 28 67 6C 65 61 6D |  0 -15 Td (gleam
 69 6E 67 3F 20 57 68 6F 73 65 20 62 72 6F 61 64 | ing? Whose broad
```

Figure 7. PDF syntax observation – restoration of text setting



I Love FCU, say can you see, by the dawn's early light, What so proudly we hailed at the twilight's last gleaming? Whose broad stripes and bright stars through the perilous fight, O'er the ramparts we watched, were so gallantly streaming? And the rocket's red glare, the bombs bursting in air,Gave proof through the night that our flag was still there. I Love FCU, say does that star-spangled banner yet wave, O'er the land of the free and the home of the brave? On the shore, dimly seen through the mists of the deep, Where the foe's haughty host in dread silence reposes, What is that which the breeze, o'er the towering steep,As it fitfully blows, half conceals, half discloses? Now it catches the gleam of the morning's first beam,In full glory reflected now shines in the stream, 'Tis the star-spangled banner! I Love FCU long may it wave O'er the land of the free and the home of the brave. And where is that band who so vauntingly swore That the

Figure 8. Altered PDF document – text altered

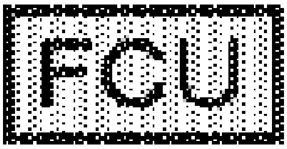


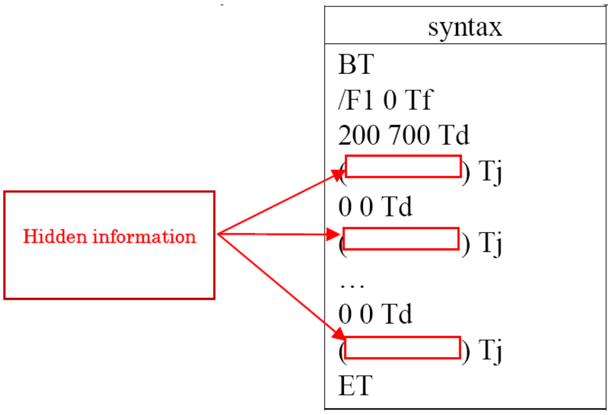
Figure 9. Retrieved watermark from the PDF file with altered text

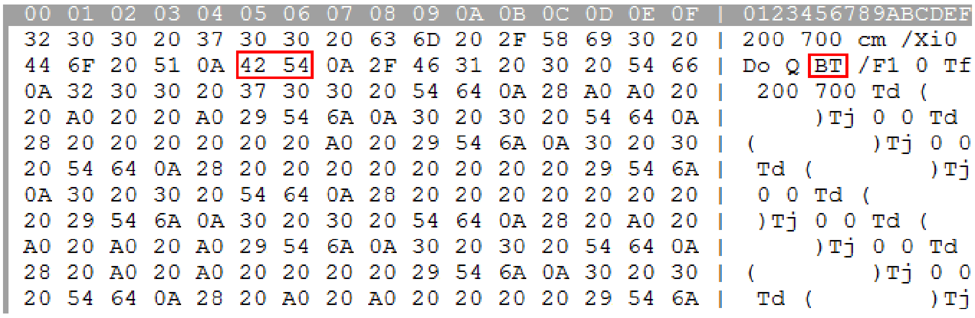
After we modified the text of the PDF file, we would like to further retrieve the watermark. The altered PDF file is shown in Figure 8, and retrieval of the watermark is shown in Figure 9.

4.2. **PDF information hiding method applied to copyright protection – Experiment 2.** We then conduct an experiment according to the method described in Section 3.6. As we return to the image of the American flag in the PDF document shown in Figure 2, the coordinates of the image in the PDF file is $(200, 700)$. We intend to hide S onto the American flag. The entire S is converted into the syntax shown in Table 7.

The segment "200 700 Td" in the syntax is enclosed between "BT" and "ET", as shown in Figures 10 and 11. Therefore, datum point for the displaying coordinate is located at the bottom left corner of the document page. The reason for setting the coordinate at $(200, 700)$ was to ensure that the first row blank and the American flag are stacked together. Because the displaying coordinate of the image is also at $(200, 700)$, and since the datum points of the displaying coordinates of the second row and after are based from the previous row; when the displaying coordinates are set as $(0, 0)$ for the second row and after, the second row would stack onto the first row, the third row blank stack onto the

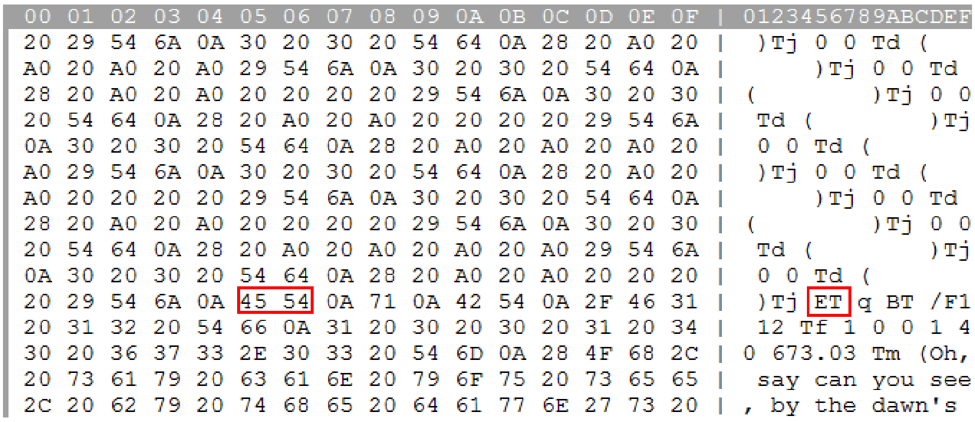
Table 7. Syntax block of verification message after conversion





```
 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F | 0123456789ABCDEF
 32 30 30 20 37 30 30 20 63 6D 20 2F 58 69 30 20 | 200 700 cm /Xi0
 44 6F 20 51 0A 42 54 0A 2F 46 31 20 30 20 54 66 | Do Q BT /F1 0 Tf
 0A 32 30 30 20 37 30 30 20 54 64 0A 28 A0 A0 20 |  200 700 Td (
 20 A0 20 20 A0 29 54 6A 0A 30 20 30 20 54 64 0A |      )Tj 0 0 Td
 28 20 20 20 20 20 20 A0 20 29 54 6A 0A 30 20 30 | (         )Tj 0 0
 20 54 64 0A 28 20 20 20 20 20 20 20 20 29 54 6A |  Td (         )Tj
 0A 30 20 30 20 54 64 0A 28 20 20 20 20 20 20 20 |  0 0 Td (
 20 29 54 6A 0A 30 20 30 20 54 64 0A 28 20 A0 20 |  )Tj 0 0 Td (
 A0 20 A0 20 A0 29 54 6A 0A 30 20 30 20 54 64 0A |      )Tj 0 0 Td
 28 20 A0 20 A0 20 20 20 20 29 54 6A 0A 30 20 30 | (         )Tj 0 0
 20 54 64 0A 28 20 A0 20 A0 20 20 20 20 29 54 6A |  Td (         )Tj
```

Figure 10. PDF syntax observation – text syntax opening



```
 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F | 0123456789ABCDEF
 20 29 54 6A 0A 30 20 30 20 54 64 0A 28 20 A0 20 |  )Tj 0 0 Td (
 A0 20 A0 20 A0 29 54 6A 0A 30 20 30 20 54 64 0A |      )Tj 0 0 Td
 28 20 A0 20 A0 20 20 20 20 29 54 6A 0A 30 20 30 | (         )Tj 0 0
 20 54 64 0A 28 20 A0 20 A0 20 20 20 20 29 54 6A |  Td (         )Tj
 0A 30 20 30 20 54 64 0A 28 20 A0 20 A0 20 A0 20 |  0 0 Td (
 A0 29 54 6A 0A 30 20 30 20 54 64 0A 28 20 A0 20 |  )Tj 0 0 Td (
 A0 20 20 20 20 29 54 6A 0A 30 20 30 20 54 64 0A |      )Tj 0 0 Td
 28 20 A0 20 A0 20 20 20 20 29 54 6A 0A 30 20 30 | (         )Tj 0 0
 20 54 64 0A 28 20 A0 20 A0 20 A0 20 A0 29 54 6A |  Td (         )Tj
 0A 30 20 30 20 54 64 0A 28 20 A0 20 A0 20 20 20 |  0 0 Td (
 20 29 54 6A 0A 45 54 0A 71 0A 42 54 0A 2F 46 31 |  )Tj ET q BT /F1
 20 31 32 20 54 66 0A 31 20 30 20 30 20 31 20 34 |  12 Tf 1 0 0 1 4
 30 20 36 37 33 2E 30 33 20 54 6D 0A 28 4F 68 2C | 0 673.03 Tm (Oh,
 20 73 61 79 20 63 61 6E 20 79 6F 75 20 73 65 65 |  say can you see
 2C 20 62 79 20 74 68 65 20 64 61 77 6E 27 73 20 | , by the dawn's
```

Figure 11. PDF syntax observation – text syntax ending

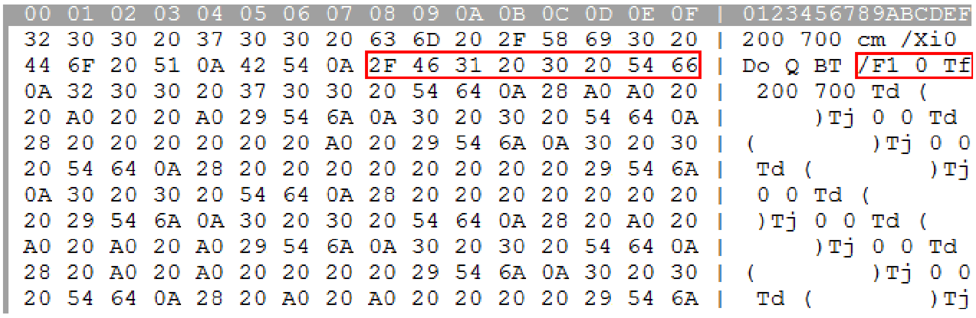
second row and so on. As a result, all of the blanks are stacked together and apply to the text setting "/F1 0 Tf", as in Figure 12.

After we modified the American flag in the PDF file, we would like to further retrieve the watermark. The altered PDF file is shown in Figure 13, and retrieval of the watermark is shown in Figure 14.

4.3. **Applications and discussions.**

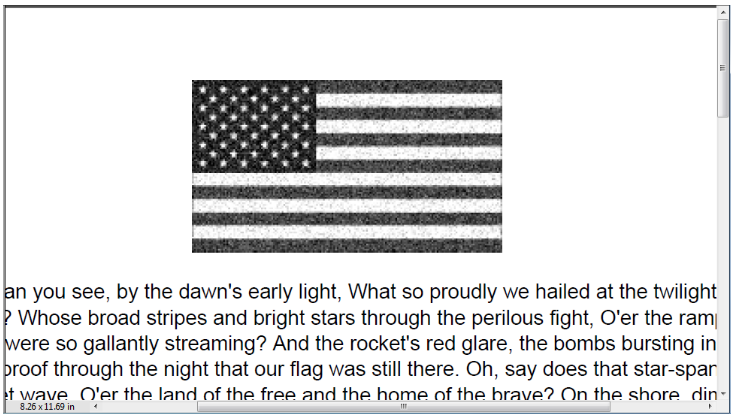
4.3.1. *Existing protection methods for a PDF file.*

(a) Protection method for PDF files. There are basically two types of protection methods for PDF documents. One is PDF security and the other is PDF encryption. The security function for PDF documents will provide the PDF document with various



```
 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F | 0123456789ABCDEF
 32 30 30 20 37 30 30 20 63 6D 20 2F 58 69 30 20 | 200 700 cm /Xi0
 44 6F 20 51 0A 42 54 0A 2F 46 31 20 30 20 54 66 | Do Q BT /F1 0 Tf
 0A 32 30 30 20 37 30 30 20 54 64 0A 28 A0 A0 20 |  200 700 Td (
 20 A0 20 20 A0 29 54 6A 0A 30 20 30 20 54 64 0A |      )Tj 0 0 Td
 28 20 20 20 20 20 20 A0 20 29 54 6A 0A 30 20 30 | (         )Tj 0 0
 20 54 64 0A 28 20 20 20 20 20 20 20 20 29 54 6A |  Td (         )Tj
 0A 30 20 30 20 54 64 0A 28 20 20 20 20 20 20 20 |  0 0 Td (
 20 29 54 6A 0A 30 20 30 20 54 64 0A 28 20 A0 20 |  )Tj 0 0 Td (
 A0 20 A0 20 A0 29 54 6A 0A 30 20 30 20 54 64 0A |      )Tj 0 0 Td
 28 20 A0 20 A0 20 20 20 20 29 54 6A 0A 30 20 30 | (         )Tj 0 0
 20 54 64 0A 28 20 A0 20 A0 20 20 20 20 29 54 6A |  Td (         )Tj
```

Figure 12. PDF syntax observation – text setting



an you see, by the dawn's early light, What so proudly we hailed at the twilight
? Whose broad stripes and bright stars through the perilous fight, O'er the ram
were so gallantly streaming? And the rocket's red glare, the bombs bursting in
proof through the night that our flag was still there. Oh, say does that star-span
t wave, O'er the land of the free and the home of the brave? On the shore, din

Figure 13. Altered PDF document – image altered

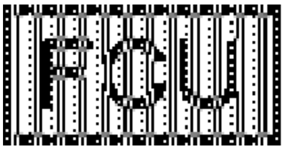


Figure 14. Retrieved watermark from the PDF file after image was altered

settings; the user is then restricted with some of the functions when reading a PDF file with PDF Reader. These restrictions include the prohibitions of printing, saving a new file and copying the text, or the context is restricted to read only, etc. The original PDF document owner is also required to set the security password when setting PDF document security. The password is used to lift the various restrictions when using PDF Reader to read a PDF file. However, the PDF document security function does not conduct encryption to the entire text. As a result, the general user can still read the PDF file using PDF Reader even if the security password is not known, only being restricted to some of the functions.

The encryption function for the PDF document will conduct encryption to the entire text. The encryption can be divided into symmetric encryption and asymmetric encryption. If the original PDF is applied with encryption by the document owner, the input of the correct password is required in order to read the PDF file with PDF Reader; otherwise the PDF file cannot be completely read.

(b) Possible attacking methods. If a person attempts to photo the PDF document into various images, and then convert all of the images into a PDF document; or if a person attempts to manually (such as re-typing) convert the PDF content into a new document, the newly fabricated PDF file will not include S.

4.3.2. *Reformed copyright protection method (particularly suitable for digital images).*
(a) Assume that the text or image of the PDF file consists of m characters (pixels) of $a_1, a_2, \ldots, a_m$, with watermark $W = w_1 \ldots w_{2t}$. We first select a suitable k value, and divide m into k parts. Let $l \equiv \left\lceil \frac{m}{k} \right\rceil$.

(1) From Remark 3.2 in Section 3.5, length $2t$ is irrelevant to the size of l.
We therefore can apply Algorithm I to watermark W and $a_1, a_2, \ldots, a_l$, and create a verification ciphertext $S_1 = (x_{k+1}, c_1, \ldots, c_{2t})$. The same method can be applied to W and $a_{l+1}, \ldots, a_{2l}$ to create verification ciphertext $S_2 = (x_{2k+1}, c_{2t+1}, \ldots, c_{4t})$. The same process is continued until we create verification ciphertext $S_k = (x_{4k+1}, c_{(4k-2)t+1}, \ldots, c_{4kt})$ from W and $a_{(k-1)l+1} \ldots a_m$.

(2) The verification ciphertexts $S_1, \ldots, S_k$ are integrated into $S = (x_{4kt+1}, c_1 c_2 \ldots c_{4kt})$.
Therefore, the PDF will consist a number of k watermarks.

(3) The PDF file, watermark, and verification ciphertext S are then sent to the copyright registration center.

(4) Publish: watermark, watermark retrieving algorithm, and public key n.

(b) If the attack method in Section 4.3.1(b) is encountered, the verification ciphertext S is removed from the PDF file. As a result, the verification S and target B (such as digital image) within the PDF file cannot be retrieved to create a watermark. The purpose is to protect the copyright of target B, rather than the PDF file or syntax of describing B in the PDF file. Algorithm IV apparently tells us that a watermark can be created as long as B and S are obtained. When a dispute of the copyright occurs, the verification ciphertext S registered at the registration center can be retrieved. The rightful owner can therefore issue copyright infringement claims to the suspicious B product.

4.3.3. *Providing customer protection for purchasing genuine products.* The copyright owner can add a “viewable” company image or text onto the PDF file, such as adding “ABC” onto “Im1” by the method in Table 8 and shown in Figure 15. The method can then prevent a stealer from using a camera to retrieve the complete image; it also allows purchasers recognize the genuine mark of a company. After a buyer purchases a PDF copy from Company ABC, the person can remove the “ABC” displaying syntax to obtain the complete image. At the same time, the purchaser can obtain the verification ciphertext S from the PDF syntax and have the public key $n = p \times q$. The purchaser can then verify the hidden watermark.

4.3.4. *Tracking piracy.* If ABC Company is about to own the copyright of a digital product, the product is coded by iso-8859-1 [2] into PDF format. The file is then inserted with a non-embedded watermark verification text, $S = (x, c)$. In this watermark text S, x can

Table 8. Adding a viewable text syntax

Original syntax	Modified syntax
q 256 0 0 256 36 550 cm /Im1 Do Q	q 256 0 0 256 36 550 cm /Im1 Do Q BT /F1 12 Tf 36 550 Td (ABC)Tj ET

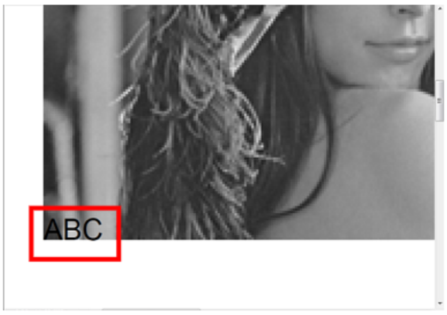


FIGURE 15. Viewable text added to the PDF document

be regarded as the product's serial number, which can be used as a secrete key to extract the watermark. The ABC Company issues a one and only unique serial number x for every digital product. The buyer's identity and the watermark verification text, $S = (x, c)$, are registered altogether and saved within the database. If the market suddenly appears with two identical serial numbers x, this indicates that somebody is illegally pirating the product. Company ABC is then able to track the source of the piracy.

4.3.5. *Discussion.* As we compare the information hiding method in this study to the approach by Lee and Tsai's method [6] as follows:

(1) Lee and Tsai paper is mainly based on unembedded information hiding; they hide data in the spacing between words of the cover text. Since there is no relationship between the hidden data and cover text, copyright owners cannot claim their rights of the text, and data cannot be hidden in image files using Lee and Tsai scheme.

(2) (i) Our scheme can protect copyrights as well as being an unembedded watermark.
(ii) Our method uses a quadratic root as the random seed to generate a series of locations within the cover object. A "$\oplus$" operation is then performed between the watermark and the corresponding character or pixel of the cover object.
(iii) Available cover objects for our scheme include text, images, software, maps, and music scores, etc.
(iv) Since the cover object is unembedded and can be reused, collisions will not occur, thus allowing us to embed large sized images or text into smaller cover objects. We summarize the statements in the following table:

Characters **Schemes**	**Data hiding**	**Integration of the watermark and the cover object**	**Cover object**	**Hiding Method**	**Claim of copyrights**
Lee and Tsai's	Yes	No	text	Unembedded	No
Our scheme	Yes	Yes	text, images, software, maps, music scores	Unembedded	Yes

5. **Conclusion and Future Work.** The text, syntax and images in PDF documents are compressed. When a PDF file is opened directly without using PDF Reader, the file would appear as a random code, thus reducing the chances of any hidden information being discovered when a PDF file is opened directly.

The method in this research basically remains two problems for applying to steganography and passing secret data. The first problem is that the file size of the PDF document would increase after hiding, due to the additional content. The second issue is that the hiding information can still be removed even if PDF document security is applied. In other words, the hidden S is removable. The next stage in further research is to study how to handle the steganographic problems in PDF documents.

REFERENCES

[1] *Adobe*, http://www.adobe.com/.

[2] *Wikipedia: ISO/IEC 8859-1*, http://en.wikipedia.org/wiki/ISO/IEC_8859-1.

[3] S. H. Wu and H. F. Lin, *A Non-Embedded Watermarking Scheme for Non-Distortion Digital Product Copyright Verification*, Master Thesis, Feng Chia University, 2004.

[4] A. Castiglione, A. D. Santis and C. Soriente, Taking advantages of a disadvantage: Digital forensics and steganography using document metadata, *The Journal of Systems and Software*, vol.80, pp.750-764, 2007.

[5] 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.

[6] I. S. Lee and W. H. Tsai, A new approach to covert communication via PDF files, *Signal Processing*, vol.90, pp.557-565, 2010.

[7] Adobe, *PDF Reference*, 6th Edition, Version 1.7, http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf.

[8] *iTextSharp*, http://sourceforge.net/projects/itextsharp/.