# Copyright Protection of E-books by Data Hiding Based on Integer Factorization

**Da-Chun Wu[1*] and Ping-Yu Hsieh[2]**
Department of Computer and Communication Engineering
National Kaohsiung University of Science and Technology
Kaohsiung 82445, Taiwan
[1] [e-mail: dcwu@nkust.edu.tw]
[2] [e-mail: kiraxie11287@gmail.com]
*Corresponding author: Da-Chun Wu

*Received April 2, 2020; revised November 6, 2020; accepted March 1, 2021; published September 30, 2021*

## Abstract

A data hiding method based on integer factorization via e-books in the EPUB format with XHTML and CSS files for copyright protection is proposed. Firstly, a fixed number $m$ of leading bits in a message are transformed into an integer which is then factorized to yield $k$ results. One of the $k$ factorizations is chosen according to the decimal value of a number $n$ of the subsequent message bits with $n$ being decided as the binary logarithm of $k$. Next, the chosen factorization, denoted as $a \times b$, is utilized to create a combined use of the <p> and  elements in the XHTML files to embed the $m + n$ message bits by including into the two elements a class selector named according to the value of $a$ as well as a text segment with $b$ characters. The class selector is created by the use of a CSS pseudo-element. The resulting web pages are of no visual difference from the original, achieving a steganographic effect. The security of the embedded message is also considered by randomizing the message bits before they are embedded. Good experimental results and comparisons with exiting methods show the feasibility of the proposed method for copyright protection of e-books.

***Keywords:*** Data Hiding, Integer Factorization, E-book, EPUB Format, XHTML File, CSS File

This work was supported partially by National Science Council, Taiwan, under the grant NSC 101-3113-P-009-006.

**http://doi.org/10.3837/tiis.2021.09.018** **ISSN : 1976-7277**

## 1. Introduction

With the rapid developments of computer networks and mobile devices, digital documents uploaded onto the Internet gradually turn to be a main source of human knowledge, among which uses of e-books have become very popular in people's daily life because of their convenience for downloading and reading anywhere at any time without paper assumption. The electronic publication (EPUB) format using the ".epub" file extension is currently one of the most widely used e-book formats. With the advance of digital processing technology, digital sources like e-books can be downloaded, copied, or modified easily by hackers [1] for illegal usages, creating serious problems of copyright infringement [2]. It is desired to have an effective way to protect the copyright of an e-book. For this aim, embedding imperceivable copyright information into the e-book by data hiding techniques [3-6] is a feasible solution. In this study, the research topic of data hiding via the e-book for the purpose of copyright protection is investigated, and a novel method based mainly on the technique of integer factorization with the secret copyright message being embedded in the e-book in the EPUB format is proposed.

An e-book in the EPUB format [7] includes an EPUB file which consists of cascading style sheet (CSS) files, extensible hypertext markup language (XHTML) files, etc. [8-9]. Although many data hiding methods via various types of multimedia have been proposed [3-6, 10-13] with some methods proposed for copyright protection of images by watermarking [4, 5], yet, there exist very few studies using e-books in the EPUB format as the cover media for data hiding. Kabetta et al. [14] proposed a data embedding method to insert spaces or tabs after the semicolon characters at the line ends in the CSS file to encode message bits. The inserted spaces or tabs are invisible on the web pages which are the browsing result of the stego-file. Lai et al. [15] proposed another data hiding method which uses different settings of the margin and padding attributes in the CSS file to encode data bits. The various ways of assigning the margin and padding attributes all generate webpages of identical appearances. Wu and Su [16] embedded message bits into the lexicographical orders of the selectors and the related declarations in the CSS files in an EPUB document. The appearance of the e-book does not change after message embedding.

On the other hand, there exist some studies on data hiding via the hypertext markup language (HTML) file that is used for web page creation. Lee and Tsai [17] proposed the use of special space codes, which appear as white spaces in the browsing result of the HTML file, to encode data bits. The method proposed by Dey, et al. [18] make alternative uses of the capital and small letters in the HTML, which are of identical meanings according to the HTML syntax, to hide data message bits of 0 and 1. But this technique is not suitable for XHTML files. Inoue, et al. [19] proposed the use of a pair of a start-tag and its matching end-tag, as well as its corresponding empty-element tag, to represent binary values for data embedding. Inoue, et al. [19] also use white spaces in the tags of HTML files to encode data bits. In addition, the order of attributes in the tag of the HTML file does not affect the effect of the tag in the browsing result of the file. This property is used by Huang et al. [20] for message bit embedding.

In the proposed method, a fixed number of the leading bits of an input binary message are used to create a series of integer factorizations from which one denoted in the form of $a \times b$ is selected. Message embedding is carried out according to the values of $a$ and $b$. Specifically, a combined use of the text-handling elements <p> and  in the XHTML files of the e-book is generated accordingly, which includes a pre-defined class selector named according to '$a$'

and a text segment with *b* characters, to accomplish the embedding of a number of message bits. The class selectors for use are created by the use of a CSS pseudo-element. A corresponding data extraction process is proposed as well. Experimental results and comparisons with other methods in the aspects of message embedding capability and resistance to text reformatting attacks show the feasibility and superiority of the proposed method. Data security has also been considered by randomizing the order of the input message bits before they are embedded.

In the remainder of this paper, a review of the structure of the e-book in the EPUB format is given in Section 2, the ideas behind the proposed method are described in Section 3, and a data embedding process and a data extraction processes which are designed to implement the proposed method are described in Section 4. Experimental results and comparisons of the proposed method with several existing methods are presented in Section 5. Finally, some concluding remarks are given in Section 6.

## 2. Review of the E-book Structure

In an e-book in the EPUB format [7], the XHTML files describe the main elements of the e-book, and the CSS files define the display properties in the XHTML files. A CSS file [8] consists of a set of *rules*, each including a *selector* and a *declaration block*, as shown in **Fig. 1** [21]. A selector is a pattern used to select the element(s) in the XHTML file to be styled. The declaration block contains one or more *declarations* separated by a semicolon, and each declaration includes a *property name* and a *value* separated by a colon (:). An example of uses of rules is as follows:

```
.sentence {text-align:center; font-weight:bold;}
```

which defines a *class selector* named `.sentence` for selecting the elements of specific class attributes in the XHTML file. This class selector can be used, e.g., by a <p> element, which defines a paragraph of text in the XHTML file, in the following way:

```
<p class="sentence">I am learning Japanese.</p>
```

to yield a centered sentence in bold on the web page as follows.

**I am learning Japanese.**

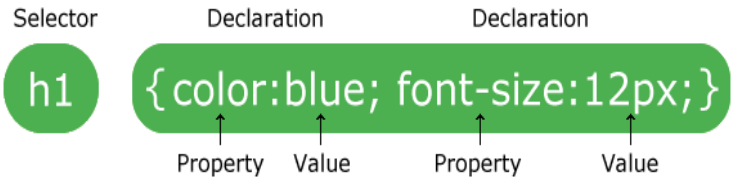


**Fig. 1.** The structure of a rule in the CSS file [21]

Next, a *pseudo-element* in a CSS file defined in the following form:

```
selector::pseudo-element{property:value;}
```

may be used to style specified parts of an element in an XHTML file. Two types of pseudo-elements provided by the CSS3 version [8], namely, `::before` and `::after`, can be used to insert some text before and after the content of an element, respectively, and either of them can be used to embed message data. For example, the following paragraph in the XHTML file:

```
p::before{content:"Beginning <";}
p::after{content:"> Ending";}
...
<p>Hi, everybody.</p>
```

will yield the following line of text on the web page.

```
Beginning <Hi, everybody.> Ending
```

Also, the  element is defined to group inline-elements in an XHTML document, by which the following *combined use* of the two elements <p> and :

```
<p>He is a <span style="font-weight:bold">tall</span> man.</p>
```

will yield the following sentence with the word ‘tall’ in bold.

```
He is a tall man.
```

In fact, the  element provides a way to add a *hook* to a part of text which can be styled with the CSS or manipulated with the JavaScript. For example, the following paragraph:

```
.use::before{content: "#"}
...
<span class="use">This is a hooked text.</span>
```

will yield the following sentence with a preceding character ‘#’:

```
#This is a hooked text.
```

where the class selector ‘`use`’ is included in the  element. It is noted that if the content specified by `::before` is changed to be *an empty string* instead of ‘#’, then the <style> element in the CSS will become *ineffective*, though *valid*.

## 3. Basic Ideas of the Proposed Method

In this section, the basic ideas of the proposed method are explained by illustrative examples, including the technique of integer factorization, the use of the pseudo-element `::before` to create class selectors in the CSS file, and a combined use of the <p> and  elements in the XHTML file to embed message bits into an e-book without changing the appearance of the e-book on the web page.

### 3.1. Embedding by Integer Factorization for Normal Cases

In number theory, integer factorization is the decomposition of a composite integer $v$ into a product of smaller integers $a$ and $b$, i.e., $v = a \times b$. In this study, we utilize all the possible integer factorization results, say $k$ ones, of a given integer $v$ described as a series of equal-valued products as follows:

$$a_1 \times b_1 = a_2 \times b_2 = \ldots = a_k \times b_k,$$

or equivalently,

$$a_1 \times (v/a_1) = a_2 \times (v/a_2) = \ldots = a_k \times (v/a_k)$$

where $a_1 < a_2 < \ldots < a_k$.

***Example 1 - Message embedding by integer factorization.***

Suppose that the message to be embedded is the bit string $S_1$ = '11010…'. Embedding of $S_1$ may be divided into two stages. In the first stage, the proposed method takes a number $m$ of leading bits of $S_1$, transforms them into a decimal value $v$, and then performs *integer factorization* to $v$ to get a series of factorization results, each in the form of a product of two integers. For example, suppose $m$ = 3 so that the three leading bits, '110', are taken from $S_1$, which, after being transformed into a decimal value, becomes $v$ = 6. Then, four factorization results, namely,

$$1 \times 6 = 2 \times 3 = 3 \times 2 = 6 \times 1,$$

can be obtained from $v$ = 6. Let the number of factorizations, four here, be denoted as $k$. These $k$ ways of factorizing $v$ are used next by the proposed method to "represent" the next $\lfloor \log_2 k \rfloor$ bits in $S$. Specifically, by denoting the value $\lfloor \log_2 k \rfloor$ by $n$ which is $n = \lfloor \log_2 4 \rfloor = 2$ bits here, it means that the four factorizations 1×6, 2×3, 3×2, and 6×1 can be used to represent the four 2-bit codes 00, 01, 10, and 11, respectively. Since the two subsequent bits of $S_1$ are '10' whose decimal value is $w$ = 2, the ($w$ + 1)-th, or equivalently, the 3rd factorization, 3×2, is chosen by the proposed method and denoted by $a \times b$.

In the second stage, firstly the proposed method creates $2^m$ *undefined* class selectors named, e.g., as '`.t0`', '`.t1`', …, '`.t(2^m-1)`', by using the pseudo-element `::before` with the associated contents being set to be an empty string. For the case here, we have $m$ = 3 and $2^m$ = 8, and so the following series of class selectors is created.

```
.t0::before{content:""}
.t1::before{content:""}
...
.t7::before{content:""}
```

Then, the leading ($m$ + $n$) = 5 bits of $S_1$, '11010', can now be embedded integrally according to the factorization $a \times b$ = 3×2 by the following combined use of the <p> and  elements:

```
<p><span class="t2">Wh</span>ere are you?</p>
```

which includes both the class selector named '`t2`' according to the value $a - 1$ (= 3 – 1 = 2) and a text segment with $b$ (= 2) characters, namely, '`Wh`', to yield the following text on the web page:

```
Where are you?
```

which is of no difference from that yielded by the following use of *just* the <p> element, meaning that a steganographic effect on the resulting web page is achieved.

```
<p>Where are you?</p>
```

In the above example, the number $m$ of leading bits taken from message $S_1$ in the first stage to provide a decimal value $v$ for integer factorization is preset to be *fixed*. But depending on the content of the taken message bits, the number $k$ of the generated factorizations is *not* and will *affect* the number $n$ of the subsequent bits taken from $S_1$ to be embedded in the second stage, as illustrated more clearly by the following example. For convenience of reference, the number $m$ will be called the *leading-bits parameter* in the sequel.

***Example 2 - Embedding based on very few factorizations.***

If the message to be embedded is $S_2$ = '01010…' and the leading-bits parameter is $m$ = 3 again, by the same reasoning as in Example 1, the decimal value of the first $m$ (= 3) bits, '010', is $v$ = 2 which can be factorized to get $k$ (= 2) factorization results, namely,

$$1\times2 = 2\times1.$$

Therefore, $n = \lfloor \log_2 k \rfloor = \lfloor \log_2 2 \rfloor = 1$, indicating that one more bit of $S_2$ should be taken out, which is just the bit '1'. So, the decimal value equivalent to this bit is $w$ = 1, implying that the ($w$ + 1)-th, or equivalently, the 2nd factorization $a\times b = 2\times1$ should be chosen. Accordingly, with $a$ = 2 and $b$ = 1, if the text for use is again '`Where are you?`', then the following XHTML elements can be used to embed the leading ($m + n$) = (3 + 1) = 4 bits of $S_2$, namely, '0101':

```
<p><span class="t1">W</span>here are you?</p>
```

where the class number '`t1`' comes from the value $a - 1 = 2 - 1 = 1$.

## 3.2. Embedding by Integer Factorization for Special Cases

In the above discussions, a paragraph $x$ defined by the <p> element is used to group $b$ characters of $x$ to accomplish the embedding work. But it might happen that the characters in $x$ are too few to satisfy the need of $b$ characters, as illustrated by the next example, where a solution to this problem is also proposed.

***Example 3 – Text characters too few for use in data hiding.***

Continuing Example 1 where the 3rd factorization $a\times b = 3\times2$ is chosen for message embedding, suppose that the text paragraph defined by the <p> element is very short now, such as just *a single character* like the '`A`' in '`<p>A</p>`.' Then, a difficulty arises because it

requires $b = 2$ characters to be grouped by the  element to accomplish the embedding work, as shown in Example 1. In this situation, a solution proposed in this study is to add an extra class selector as follows to the original eight ones mentioned above.

```
.t8::before{content:""}
```

Then, the subsequent paragraph defined by *another* <p> element in the XHTML file is used to provide the characters for use by a subsequent  element to group the remaining characters *not* completed yet by the first <p> element. For example, assume that the subsequent <p> element is as follows.

```
<p>You are kind.</p>
```

Now, embedding of the five bits '110' and '10' of $S_1$ mentioned in Example 1 can be implemented by the following double combined use of the <p> and  elements.

```
<p><span class="t2">A</span></p>
<p><span class="t8">Y</span>ou are kind.</p>
```

In other words, the second  element with the extra-added class selector named '*t*8' works as an "indicator" that the character 'Y' grouped by it is a continuation of the character 'A' to yield the two characters 'AY' required by $b = 2$ in $a \times b = 3 \times 2$.

The last example illustrates the situation of insufficient text characters in a paragraph for use by the  element for message embedding. This situation might occur as well when an XHTML *file* has inadequate characters for use by  elements. In that case, a scheme similar to that used in the last example by adding an extra class selector for use by a  element *at the beginning of a second file* may be applied. The two schemes mentioned above will be called *across-paragraph embedding* (abbreviated as *APE*) and *across-file embedding* (abbreviated *AFE*), respectively, in the subsequent discussions.

Furthermore, it is possible that the leading $m$ bits taken from the message are all 0's. Then, the resulting integer factorization operation becomes *meaningless* as can be Fig.d out from the data embedding process demonstrated in the last examples. For this case, a solution proposed in this study is illustrated by the following example.

***Example 4 – Dealing with leading message bits which are all 0's.***

Suppose that the message to be embedded this time is the bit string $S_4$ = '<u>000</u>10…' and that the leading-bits parameter $m$ mentioned in the last two examples is still taken to be $m = 3$. Therefore, the leading three bits in $S_4$, '000', are taken out and transformed into a decimal value $v = 0$. The factorization of $v = 0$ becomes meaningless and message-bit embedding cannot be continued.

A solution to this problem proposed in this study is to create *artificially* $2^m$ *virtual factorizations*:

$$1 \times 0 = 2 \times 0 = \ldots = 2^m \times 0,$$

and the desired factorization $a \times b$ is chosen as in the normal case of $v \neq 0$, leading to the selection of the $(w + 1)$-th factorization $a \times b = (w + 1) \times 0$ where $w$ is the decimal value of the

subsequent *m* bits taken from the message. This means that *no* character will be grouped by the  element, though the class selector named '`t(a–1)`' (= '`tw`') is included.

Finally, it is noted that if there is a character entity [9] which starts with '&' and ends with ';' (e.g., ' '), a sequence of space characters, or any other element nested within the <p> element, then in order to avoid *breaking* such an entity or element to yield an erroneous web page browsing result, the proposed method treats the entire text of such an entity, sequence, or element *as a single character* when counting the number of characters in the message embedding and extraction processes.

## 4. Proposed Embedding and Extraction Processes

With the ideas of the proposed method illustrated by the above examples, it is now ready to present the proposed data embedding and extraction processes which implements the proposed method.

### 4.1. Proposed Data Embedding Algorithm

The proposed data embedding algorithm as described in the following includes three major steps: initialization, message-bit embedding, and ending.

***Algorithm 1 - Data embedding by integer factorization.***

**Input.** The EPUB file $E$ of an e-book, a binary message $S$ with $t$ bits, a random number generator $G$ controlled by a secret key $K$, and a leading-bits parameter $m$.

**Output.** A stego-EPUB file of the e-book with $S$ embedded.

**Steps.**

1. *(Initialization)* Perform Steps 1.1 through 1.3 below as initialization steps.
   - 1.1 Decompose EPUB file $E$ to obtain a set $C$ of CSS files and a set $X$ of XHTML files, $X_1, X_2, \ldots, X_N$, where each $X_p$ with $p = 1, 2, \ldots, N$ has $n_p$ text paragraphs, $x_{p1}, x_{p2}, \ldots, x_{pn_p}$, and each paragraph $x_{pq}$ with $q = 1, 2, \ldots, n_p$ has a sequence of $r_{pq}$ characters, '$c_1c_2\ldots c_{rpq}$', enclosed by a <p> element in the following form.

     `<p>`$c_1c_2\ldots c_{rpq}$`</p>`

   - 1.2 Create $2^m+1$ *undefined* class selectors in $C$ using the pseudo-element `::before` with the content of an empty string, where each class selector is of the following form:

     ```
     .ti::before{content:""}
     ```

     where "`ti`" is the name of the $(i + 1)$-th selector for $i = 0, 1, \ldots, 2^m$, and the last selector named '$2^m$' is defined for use as an indication of APE or AFE; also, add all the created class selectors to *each* CSS file in $C$, resulting in a new set $C'$ of CSS files.
   - 1.3 Randomize the bits in the input message $S$ using the random number generator $G$ controlled by the input key $K$ to form a new bit sequence $S'$ with $t$ bits; and if $t < m$, then append 0's to the end of $S'$ to make $t = m$.
2. *(Embedding message bits)* Take the first XHTML file and its first text paragraph by setting $p = 1$, $q = 1$; define $x_{pq}' = x_{pq}$, $r = r_{pq}$; and perform Steps 2.1 through 2.5 below till all the $t$ bits in $S'$ is embedded.

2.1 Take the leading $m$ bits away from $S'$, transform them into a decimal number $v$, and perform one of the following two cases.
(a) If $v \neq 0$, find all the integer factorizations of $v$:

$$a_1 \times (v/a_1) = a_2 \times (v/a_2) = \ldots = a_k \times (v/a_k),$$

where $0 < a_1 < a_2 < \ldots < a_k < 2^m$ and $v/a_i$ is an integer for $i = 1, 2, \ldots, k$.
(b) If $v = 0$, create artificially $2^m$ virtual factorizations:

$$1 \times 0 = 2 \times 0 = \ldots = 2^m \times 0;$$

and set $k = 2^m$.

2.2 Let $n = \lfloor \log_2 k \rfloor$; if $t < n$, then append 0's to the end of $S'$ to make $t = n$; take out the leading $n$ bits from $S'$ and update $t$ by $t = t - n$; transform the $n$ bits into a decimal value $w$; take the $(w+1)$-th factorization from the $k$ ones derived in Step 2.1 above; and denote it as $a \times b$ which can be Fig.d out to be

$$a \times b = a_{(w+1)} \times [v/a_{(w+1)}] \quad \text{if } v \neq 0, \text{ or}$$
$$= (w + 1) \times 0 \quad \text{if } v = 0.$$

2.3 Perform either case below.
(a) If $r < b$, insert into paragraph $x_{pq}'$ a new  element with a class selector named '`t(a–1)`' as follows:

```
<p>y<span class="t(a–1)">x</span></p>
```

where $x$ denotes $r$ *unused* characters and $y$, initially empty, denotes those already used for data embedding, both of $x_{pq}'$; also, set $b = b - r$ and go to Step 2.4.
(b) If $r \geq b$, do the same as in (a) except that $x$ there denotes the leading $b$ characters taken out of $x_{pq}'$; also, set $r = r - b$, and go to Step 2.5.

2.4 *(Operations for APE or AFE)* Perform the following steps.
(a) If $q = n_p$, then take a new XHTML file by setting $p = p + 1$, $q = 1$, $x_{pq}' = x_{pq}$, $r = r_{pq}$ for AFE; else, take a new text paragraph by setting $q = q + 1$, $x_{pq}' = x_{pq}$, $r = r_{pq}$ for APE.
(b) Perform either case below.
(i) If $r < b$, insert into $x_{pq}'$ a new  element with a class selector named '`t(2`$^m$`)`' as follows:

```
<p><span class="t(2m)">x</span></p>
```

where $x$ denotes all the characters of $x_{pq}'$; also, set $b = b - r$ and go to Step 2.4 to perform the step recursively.
(ii) If $r \geq b$, do the same as in (i) except that $x$ there denotes the leading $b$ characters taken out of $x_{pq}'$; also, set $r = r - b$, and continue to perform Step 2.5.

2.5 Perform either case below.
(a) If $t = 0$, i.e., if no message bit remains, go to Step 3.

(b) If $t > 0$, then if $r > 0$, then go to Step 2.1; else, perform the operations below and go to Step 2.1:

if $q = n_p$, then
take the next unprocessed XHTML file and its first text paragraph
by setting $p = p + 1$, $q = 1$, $x_{pq}' = x_{pq}$, $r = r_{pq}$;
else,
take the next unprocessed text paragraph
by setting $q = q + 1$, $x_{pq}' = x_{pq}$, $r = r_{pq}$.

3. *(Ending)* Compose the resulting set $C'$ of CSS files, the resulting set $X'$ of XHTML files, and other untouched files in the original EPUB file $E$, to form the desired stego-EPUB file $E'$ and exit.

## 4.2. Proposed Data Extraction Algorithm

Similar to the proposed data embedding algorithm, the proposed data extraction algorithm as described in the following includes three major steps: initialization, message-bit extraction, and ending.

***Algorithm 2 - Data extraction.***

**Input.** A stego-EPUB file $E'$ of an e-book in which a random-bit message sequence $S$ is embedded and $2^m+1$ class selectors as those created in Algorithm 1 are included; the leading-bits parameter $m$, the random number generator $G$, and the secret key $K$ used in Algorithm 1.

**Output.** The message $S$ with $t$ bits embedded in $E'$.

**Steps.**

1. *(Initialization)* Decompose stego-EPUB file $E'$ to obtain a set $C'$ of CSS files and a set $X'$ of XHTML files, $X_1', X_2', \ldots, X_N'$, where each $X_p'$ with $p = 1, 2, \ldots, N$ has $n_p$ text paragraphs, $x_{p1}', x_{p2}', \ldots, x_{pn_p}'$.
2. *(Extracting message bits)* Let $S'$ be a bit sequence, preset empty initially and let $t'$ be a counter, preset zero initially; take the first XHTML file and its first text paragraph by setting $p = 1$ and $q = 1$; and perform Steps 2.1 through 2.5 below until all the XHTML files have been checked, i.e., until $p = N$ and $q = n_p$.
   2.1 Scan $x_{pq}'$ to find the first *unprocessed* combined use of the two elements <p> and  with a class selector named '$ta'$' in the following form:

```
<p>y<span class="ta'">x</span>…</p>
```

   where $x$ denotes $b$ characters, and $y$, initially empty, denotes *processed* characters, both of $x_{pq}'$; and if this is the *last* combined use of <p> and  in $x_{pq}'$, then perform the next step, Step 2.2; else, go to Step 2.4.
   2.2 If $q = n_p$, then take the next *unprocessed* XHTML file and its first text paragraph by setting $p = p + 1$ and $q = 1$; else, take the next *unprocessed* text paragraph by setting $q = q + 1$; scan $x_{pq}'$ to find the first combined use of the <p> and  elements: if it is in the following form which includes the class selector named '$t(2^m)$':

`<p><span class="t(2^m)">x'</span>…</p>`

where $x'$ denotes $b'$ characters, then regard the class selector named '`t(2^m)`' as an indicator for AFE or APE and continue to perform Step 2.3; else, go to Step 2.4.

2.3 Renew the value $b$ by setting $b = b + b'$ where $b'$ is found in Step 2.2; also, if the combined use of <p> and  just found in Step 2.2 is the last one in $x_{pq}'$, then go to Step 2.2; else, continue to perform Step 2.4.

2.4 Let $a$ be the value of $a'+1$ and let $s'$ be the value of the product $a \times b$; apply integer factorization to $s'$ to obtain $k$ factorizations of $s'$ in a way as described in Step 2.1 of Algorithm 1; compute $n = \lfloor \log_2 k \rfloor$; and let the factorization $a \times b$ be the $w'$-th one in the $k$ factorizations of $s'$.

2.5 Transform $s'$ into an $m$-bit binary string, transform $(w' - 1)$ into an $n$-bit binary string $s''$, replace $S'$ by the new bit string $S' = S' \cdot s' \cdot s''$ where the operation '$\cdot$' means bit concatenation, and update the counter value $t'$ by $t' = t' + m + n$; and if $t' \geq t$ (i.e., if all message bits have been extracted), then continue to perform Step 3; else, go to Step 2.1.

3. *(Ending)* Take the leading $t$ bits of $S'$ to form a bit sequence, de-randomize the sequence by the random number generator $G$ using the secret key $K$, and take the result as the desired message $S$ and exit.

## 5. Experimental Results and Comparisons with Other Methods

In this section, the data obtained by the experiments conducted in this study are analyzed, with focuses on some interesting trends shown by the graphs of the data. Then, some intermediate computation results are shown to verify the correctness of the experimental results yielded by the proposed data embedding algorithm (Algorithm 1) and data extraction algorithm (Algorithm 2). Finally, comparisons of the proposed method with five existing related methods are conducted to show the superiority of the proposed method.

### 5.1. Analysis of Experimental Data

To verify the correctness of the proposed data embedding and extraction algorithms (Algorithms 1 and 2), a series of experiments have been conducted on the EPUB files of 20 e-books collected from websites [22]. Under the *fair* condition that the input message is an *identical random* bit sequence which is long enough for each of the 20 EPUB files, the data embedding capacity of each file was counted and some parameters related to the embedded data are computed. The results are presented in **Table 1** by numbers and in **Figs. 2 through 4** by graphs.

**Table 1** shows the original sizes of the e-books in the unit of byte and the total number of characters enclosed by all the <p> elements in the EPUB file of each e-book, as well as the average values of these data. **Figs. 2 through 4** show respectively the graphs of three parameters related to data embedding results, namely, the *bit embedding capacity*, the *number of embedded bits per character* enclosed by the <p> elements, and *the size of the EPUB file* of each e-book, all yielded after data embedding for different values (ranging from 3 through 15) of the leading-bits parameter $m$ used in the data embedding and extraction algorithms. The reasons for the trends of these graphs are explained in the following.

### 5.1.1. Analysis of Bit Embedding Capacity

From the illustration of **Fig. 2**, the following two phenomena can be observed:

(F1) the larger the leading-bits parameter value $m$ is, the smaller the embedding capacity becomes; and

(F2) when the parameter $m$ becomes large up to be about 15, the bit embedding capacity becomes very small relatively (specifically the *average* embedding capacity of all the files reduces from 807565 bits for $m = 3$ down to 1736 bits for $m = 15$ with a reduction rate of 1/465).

The reason for the phenomena of (F1) and (F2) to occur is described in the following:

(1) as $m$ approaches large, the value $b$ in the chosen factorization $a \times b$ becomes large as well so that the characters enclosed in the <p> elements are *exhausted quickly*;

(2) however, the number of embedded bits, $m + n$, is still *small* because $m$ is fixed and $n$ is computed as $\lfloor \log_2 k \rfloor$ which can be Fig.d out to be no larger than $m$ and so is not large (note that $k$ is the number of generated factorizations);

(3) as a result, when $m$ becomes very large, only a small number of message bits are embedded before the characters in the e-book are exhausted.

**Table 1.** EPUB file sizes of the tested e-books

| Name of e-book | File size before data embedding (byte) | Number of characters in <p> elements for data embedding |
|---|---|---|
| B01 (The Best Ghost Stories) | 206,301 | 439,116 |
| B02 (Rain on My Wings) | 280,816 | 37,425 |
| B03 (Favorite Fairy Tales) | 2,642,362 | 207,302 |
| B04 (Hans Andersen's Fairy Tales) | 860,248 | 344,621 |
| B05 (The Arabian Nights Entertainment) | 993,921 | 466,440 |
| … | … | … |
| Average | 698,397 | 416,523 |

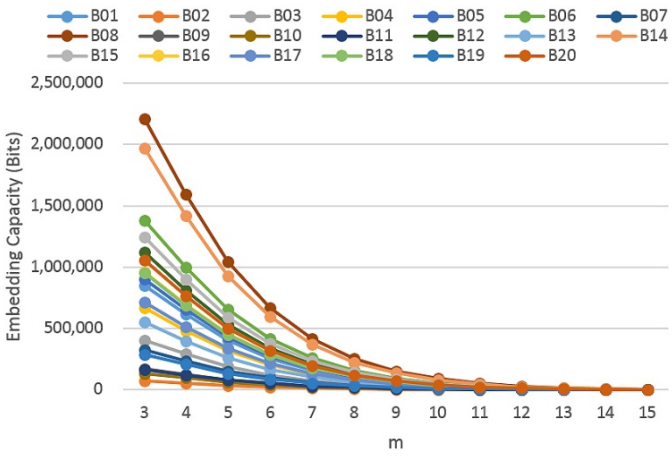


**Fig. 2.** An illustration of the bit embedding capacities of the 20 tested EPUB files yielded after data embedding using different values of the leading-bits parameter $m$

### 5.1.2. Analysis of the Numbers of the Embedded Bits per Character

It is interesting to see the following two facts from **Fig. 3** which shows the number of embedded bits per character enclosed by the <p> elements in the EPUB file of each e-book:

(F3) the number of embedded bits per character becomes small as the leading-bits parameter *m* becomes large (quite similar to the trend of **Fig. 2**);

(F4) while the embedding capacities of the files are different, these numbers of embedded bits per character of all the e-books are *almost identical*.

The reason why the phenomena of (F3) and (F4) occur is described in the following:

(1) the phenomenon of (F3) is just a *direct* consequence of the phenomenon of (F1) as can be Fig.d out;

(2) the input message is random, identical, and long enough for each e-book so that the embedding of the message can be conducted to reach the end of the e-book;

(3) for each value of the leading-bits parameter *m*, since the proposed message-bit embedding algorithm carries out *the same* steps to process the text in each e-book according to the bits in an *identical* input message, the computation result of the number of embedded bits per character theoretically should be the same for each e-book as long as the value of m is fixed;

(4) however, because the characters in the EPUB files for message-bit embedding are different in number and in distribution, the message bits embedded into the EPUB files will cause different embedding results at the end of the embedding process (possibly consuming different numbers of message bits at the end of the embedding algorithm), leading to creation of small variations among the computed numbers of embedded bits per character of the e-books, as shown by the enlargement of a part of the curve (between $m = 5$ and $m = 6$) in **Fig. 3**.

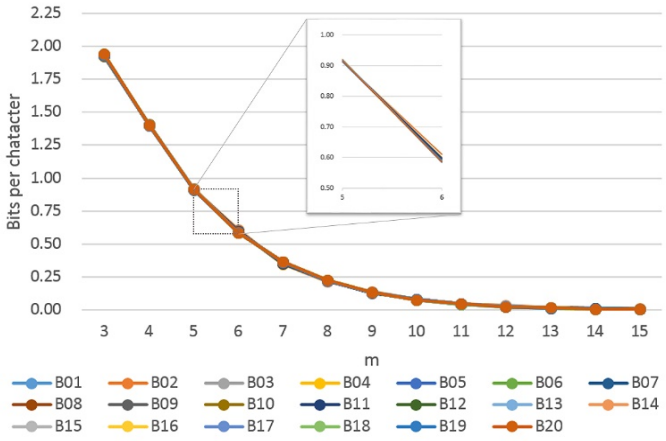


**Fig. 3.** An illustration of the numbers of embedded bits per character of the 20 tested EPUB files yielded after data embedding using different values of the leading-bits parameter *m*

### 5.1.3. Analysis of the Differences of File Sizes after Data Embedding

**Fig. 4** shows the trends of the differences of the EPUB file sizes from their original ones after data embedding is carried out, from which it is interesting to observe the following facts:

(F5) each of the differences of the EPUB file sizes from their original ones decreases to a minimum as *m* increases, and then increases again; and

(F6) the minimum values occur roughly in the range from $m = 9$ to $m = 12$.

The reason for (F5) and (F6) is described in the following:

(1) as $m$ is very small, the created $b$ value in the selected integer factorization $a \times b$ is also small so that embedding of the message will divide the text paragraphs enclosed by the <p> elements into small segments with the extra data of many class selectors being included, leading to an increase of the resulting file-size differences;

(2) on the other hand, as $m$ becomes large (larger than 12 roughly), it needs a lot of text paragraphs to embed the message with many class selectors with the name '$t(2^m)$' being included in the meantime, leading as well to an increase of the file-size difference.

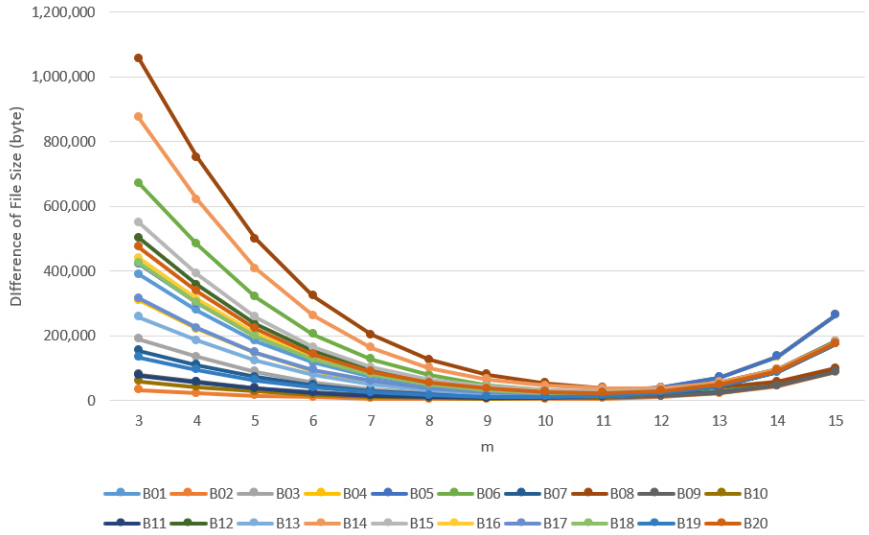


**Fig. 4.** An illustration of differences of the file sizes of the 20 tested EPUB files from their original ones, which are yielded after data embedding using different values of the leading-bits parameter $m$

## 5.2. Verifying the Correctness of Experimental Results

In order to verify the correctness of the proposed data embedding and extraction algorithms (Algorithms 1 and 2), some intermediate computation results are shown to verify the correctness of the experimental results yielded by the proposed data embedding algorithm (Algorithm 1) and the data extraction algorithm (Algorithm 2).

### 5.2.1 Verifying the Correctness of Data Embedding

**Fig. 5** shows the appearances of part of an XHTML file of the e-book "The Best Ghost Stories" before and after the bits of the message "I love you" are embedded with the leading-bits parameter taken to be $m = 5$. For convenience of verification of the results shown in the Fig., the message bits were not randomized before embedding. Also, **Table 2** is generated to show part of the *intermediate* results of carrying out Algorithm 1 to embed this message. These results correspond to the first two lines of text in **Fig. 5**(b), showing partially the correctness of the performance of the algorithm.

```
<p>The Project Gutenberg eBook, The Best Ghost Stories, by Various</p>
<div class="pgmonospaced pgheader">This eBook is for the use of anyone
anywhere at no cost and with<br/>almost no restrictions whatsoever.
You may copy it, give it away or<br/>re-use it under the terms of the
Project Gutenberg License included<br/>with this eBook or online at <a
tag="{http://www.w3.org/1999/xhtml}a">www.gutenberg.org</a></div>
<p>Title: The Best Ghost Stories</p>
```

(a)

```
<p><span class="t0">The Proje</span><span class="t0">ct Gutenb
</span><span class="t27"></span><span class="t0">erg</span><span class=
"t29"> </span><span class="t3">eBo</span><span class="t0">ok, T
</span><span class="t0">he Best </span><span class="t0">Ghost Stories,
</span><span class="t21">b</span><span class="t8">y V</span><span class=
"t0">arious</span></p>
<div class="pgmonospaced pgheader">This eBook is for the use of anyone
anywhere at no cost and with<br/>almost no restrictions whatsoever.
You may copy it, give it away or<br/>re-use it under the terms of the
Project Gutenberg License included<br/>with this eBook or online at <a
tag="{http://www.w3.org/1999/xhtml}a">www.gutenberg.org</a></div>
<p><span class="t32">Title: The Bes</span>t Ghost Stories</p>
```

(b)

**Fig. 5.** Part of the XHTML file of the e-book “The Best Ghost Stories” (downloaded from Project Gutenberg [23]) before and after embedding message “I love you.” (a) Before message embedding. (b) After message embedding with $m = 5$

**Table 2.** Part of the intermediate results of embedding the message “I love you” corresponding to the first two lines of text in **Fig. 3**(b) (notes: the bit string of the message “I love you” is 01001001001000000110110001l0…; the symbols used in the table are identical to those used in Algorithms 1)

| Column No. | (1) | (2) | (3) | (4) | (5) | (6) | (7) |
|---|---|---|---|---|---|---|---|
| Row No. | Decimal value $v$ of $m$ message bits ($m = 5$) | No. $k$ of integer factorization of $v$ (List of integer factorizations) | $n = \lfloor \log_2 k \rfloor$ (Serial no. $2^n$ of selected $a \times b$ in integer factorization Sequence) | Sequence $s$ of next $n$ message bits | Decimal value $w$ of $s$ | Selected ($w$+1)-th integer factorization $a \times b$ | Embedding ($m + n$) bits into: $y$$x$… ($x$: leading characters in the message) |
| 1 | $v = 9_{10} = 01001_2$ | $k = 3$ (9=1×9=3×3=9×1) | $n = \lfloor \log_2 3 \rfloor = 1$ | $s = 0$ | $w = 0_{10} = 0_2$ | $a \times b = 1 \times 9$ | The Proje |
| 2 | $v = 9_{10} = 01001_2$ | $k$=3 (9=1×9=3×3=9×1) | $n = \lfloor \log_2 3 \rfloor = 1$ | $s = 0$ | $w = 0_{10} = 0_2$ | $a \times b = 1 \times 9$ | ct Gutenb |
| 3 | $v = 0_{10}=00000_2$ | $k=2^m=2^5=32$ (32 virtual factor. 1×0=2×0=…=32×0) | $n = \lfloor \log_2 32 \rfloor = 5$ | $s = 11011$ | $w = 27_{10} = 11011_2$ | $a \times b = 28 \times 0$ |  |
| 4 | $v = 3_{10} = 00011_2$ | $k$=2 (3=1×3=3×1) | $n = \lfloor \log_2 2 \rfloor = 1$ | $s = 0$ | $w = 0_{10} = 0_2$ | $a \times b = 1 \times 3$ | erg… |

As a demonstration, with the input message $S$ being ‘I love you’ some whose leading bits are ‘01001$\underline{0}$010010…’ and the leading-bits parameter being $m = 5$, some intermediate

computational results for the first five message bits '01001' yielded by Algorithm 1 are as shown in the first row of **Table 2** and listed in the following for a detailed check:

(1) $v = 9_{10} = 01001_2$ (five leading bits of $S$);
(2) $k = 3$ ($v$ can be factorized to be: $1\times9 = 3\times3 = 9\times1$);
(3) $n = \lfloor \log_2 k \rfloor = \lfloor \log_2 3 \rfloor = 1$;
(4) $s$ = next $n$ leading bits in $S = 0_2$;
(5) $w$ = decimal value of $s = 0$;
(6) ($w$ + 1)-th integer factorization $a\times b$ = the first $a\times b = 1\times9$;
(7) message-bit embedding by use of the  and  elements in terms of the following formula:

*y*<span class="`t(a–1)`">*x*

yields the following partial result

<span class="`t0`">The Proje…

where $x$ = 'The Proje' is the leading $b$ (= 9) characters of those enclosed by the first  element in the e-book (see **Fig. 5**(a)); and `t(a–1)` = `t0` is the class selector corresponding to $a$.

The above result can be seen in the first line in **Fig. 5**(b), proving partially that the result in **Fig. 5**(b) is correct.

**Fig. 6** shows the appearances of the contents of the corresponding e-book pages before and after message embedding as viewed through an e-book reader. The two pages in **Fig. 6**(a) and **Fig. 6**(b) can be seen to look identical to each other, showing that the desired steganographic effect has been achieved by the proposed data embedding algorithm (Algorithm 1).

### 5.2.2 Verifying the Correctness of Data Extraction

**Table 3** is generated additionally to show part of the intermediate results of running Algorithm 2 to extract the message from **Fig. 5**(b). These results correspond to the first two lines of text in **Fig. 5**(b), showing partially the correctness of the performance of the algorithm.

As a demonstration, with the leading-bits parameter being $m = 5$ and the output message bit string being '01001<u>0</u>01001<u>0</u>…' whose text is $S$ = 'I love you', some intermediate computational results for the first five output message bits '01001' yielded by Algorithm 2 are as shown in the first row of **Table 3** and listed in the following for a detailed check:

(1) message-bit extraction by use of the  and  elements in terms of the following formula:

*y*<span class="`t(a′)`">*x*

yields the following partial result

<span class="`t0`">The Proje⋯

where $x$ = 'The Proje' is the text segment with $b$ (= 9) characters which is enclosed by the first <p> element in the e-book (see **Fig. 5**(b)); and `t(a′)= t0` is the class selector corresponding to $a'$(= 0).

(2) $a \times b = 1 \times 9$ ($a = a' + 1$);
(3) $s' = 9_{10} = 01001_2$ (the first five output message bits);
(4) $k = 3$ ($s'$ can be factorized to be: $1 \times 9 = 3 \times 3 = 9 \times 1$);
(5) $n = \lfloor \log_2 k \rfloor = \lfloor \log_2 3 \rfloor = 1$ (factorized $s'$ can be selected to be: $1 \times 9 = 3 \times 3$);
(6) $w' = 1$ ($1 \times 9 = w'$-th integer factorization $s'$ = the first $a \times b$);
(7) $s'' = 0_2$ (the next $n$ output message bits).

The above result can be extracted from the first line in **Fig. 5**(b), proving partially that the result extracted from **Fig. 5**(b) is correct.


**Fig. 6.** Appearances of the web page of the e-book "The Best Ghost Stories" (downloaded from Project Gutenberg [23]) before and after message "I love you" embedding. (a) Appearance before message embedding, (b) Appearance after message embedding with $m$ = 5 (bits)

### 5.3. Comparisons with Other Methods

Three comparisons of the proposed method with five others have also been conducted in this study from the viewpoints of (1) data embedding capacity, (2) file-size difference and file-size change rate, and (3) resistance to text reformatting attacks, respectively. About the first comparison, the resulting numbers of message bits embedded in five of the 20 tested e-books using the five methods and the proposed one with $m$ = 3 and $m$ = 9 are listed in **Table 4**. Note that in the table, the two methods proposed in Inoue, et al. [19] are labeled as [19](a) as [19](b),

respectively (one using the so-called empty element and the other using the white spaces in the tag, for message embedding). As can be seen from comparison result listed in the table, the proposed method can embed much more data than the other five methods when $m$ = 3. Also, although as the value of $m$ increases, the data embedding capacity yielded by the proposed method decreases, yet even when $m$ = 9, the capacity yielded by the proposed method decreases, yet even when $m$ = 9, the capacity yielded by the proposed method is still larger than four of the other methods; it is smaller only than that yielded by Lee and Tsai [17].

**Table 3.** Part of the intermediate results of extracted message bit string "0100100100100000011011000110…" corresponding to the first two lines of text in **Fig. 3**(b) (notes: the bit string "0100100100100000011011000110" represents the message "I love you"; the symbols used in the table are identical to those used in Algorithms 2)

| Column No. | (1) | (2) | (3) | (4) | (5) | (6) | (7) |
|---|---|---|---|---|---|---|---|
| Row No. | Deciding values of $a'$ and $b$ from $y$$x$… ($x$: $b$ characters in the span element) | Forming integer factorization $a \times b$ from (1) ($a=a'+1$) | Extracting $m$ message bits ($m$ = 5) by transforming value $s'$ into $m$-bit binary number ($s'=a \times b$) | No. $k$ of integer factorization of $s'$ (List of integer factorizations) | $n = \lfloor \log_2 k \rfloor$ (Serial no. $2^n$ of selected $a \times b$ in integer factorization sequence) | Taking integer factorization $a \times b$ of (2) be the $w'$-th one in integer factorization sequence *of* (5) | Extracting next $n$ message bits $s''$ by transforming value $w'$-1 into $n$-bit binary number |
| 1 | The Proje | $a \times b = 1 \times 9$ | $s' = 9_{10} = 01001_2$ | $k$ = 3 (9 = 1×9 = 3×3 = 9×1) | $n = \lfloor \log_2 3 \rfloor = 1$ (1×9 = 3×3) | $w' = 1$ | $s'' = 0$ |
| 2 | ct Gutenb | $a \times b = 1 \times 9$ | $s' = 9_{10} = 01001_2$ | $k$ = 3 (9 = 1×9 = 3×3 = 9×1) | $n = \lfloor \log_2 3 \rfloor = 1$ (1×9 = 3×3) | $w' = 1$ | $s'' = 0$ |
| 3 |  | $a \times b = 28 \times 0$ | $s' = 0_{10} = 00000_2$ | $k = 2^m = 2^5 = 32$ (32 virtual factor. 1×0 = 2×0 = … = 32×0) | $n = \lfloor \log_2 32 \rfloor = 5$ (32 virtual factor. 1×0 = 2×0 = … = 32×0) | $w' = 28$ | $s'' = 11011$ |
| 4 | erg… | $a \times b = 1 \times 3$ | $s' = 3_{10} = 00011_2$ | $k$ = 2 (3 = 1×3 = 3×1) | $n = \lfloor \log_2 2 \rfloor = 1$ (1×3 = 3×1) | $w' = 1$ | $s'' = 0$ |

**Table 4.** A comparison of the data embedding capacities in the unit of bit of five existing methods and the proposed method for message embedding

| **E-book title** | **Kabetta et al. [14]** | **Lee & Tsai [17]** | **Inoue et al. [19](a)** | **Inoue et al. [19](b)** | **Huang et al. [20]** | **Proposed method** | |
|---|---|---|---|---|---|---|---|
| | | | | | | ***m*=3** | ***m*=9** |
| B01 | 7,307 | 248,412 | 533 | 4,717 | 829 | 698,224 | 64,528 |
| B02 | 234 | 22,377 | 33 | 544 | 36 | 59,156 | 5,563 |
| B03 | 4,569 | 126,624 | 738 | 3,918 | 1,744 | 401,500 | 27,488 |
| B04 | 6,400 | 212,112 | 670 | 4,552 | 1,528 | 667,800 | 46,848 |
| B05 | 8,013 | 266,262 | 536 | 5,200 | 1,423 | 905,307 | 62,222 |

About the second comparison, the file-size differences and the file-size change rates of the five existing methods and the proposed method after message embedding are shown in **Table 5**. The resulting values of file-size differences and the file-size change rates after embedding messages in five of the 20 tested e-books using the five methods and the proposed one with $m$

= 3 and $m$ = 9 are listed in **Table 5**. The file-size change rate is defined to be the ratio of the file-size change after data embedding (in the unit of byte) to the number of embedding bits. As can be seen from the comparison result listed in **Table 5**, the proposed method sometimes (i.e., for some $m$ values) yields large file-size differences which are larger than those yielded by the other five methods. However, this is because the proposed method in general embeds more data than the other methods; furthermore, the file-size change rate of the proposed method is better (i.e., smaller) than those yielded by most of the other methods (i.e., than the methods of [14], [17], and [19](a)) for most values of $m$. One exception here is Huang et al. [20] which does not create file-size difference and so has zero file-size change rates. However, the embedding capacities of the proposed method are much larger than those of Huang et al. [20] as can be seen from **Table 4**.

**Table 5.** A comparison of the file-size differences and file-size change rates of five existing methods and the proposed method after message embedding

| E-book title | Comparisons | Kabetta et al. [14] | Lee & Tsai [17] | Inoue et al. [19](a) | Inoue et al. [19](b) | Huang et al. [20] | Proposed method | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | $m$=3 | $m$=9 |
| B01 | File-size difference (byte) | 7,307 | 310,045 | 1,565 | 2,317 | 0 | 390,456 | 31,510 |
| | File-size change rate | 1.000 | 1.248 | 2.936 | 0.491 | 0.000 | 0.559 | 0.488 |
| B02 | File-size difference (byte) | 234 | 27,941 | 118 | 294 | 0 | 32,154 | 3,890 |
| | File-size change rate | 1.000 | 1.249 | 3.576 | 0.540 | 0.000 | 0.544 | 0.699 |
| B03 | File-size difference (byte) | 4,569 | 157,949 | 2,354 | 1,931 | 0 | 189,671 | 17,091 |
| | File-size change rate | 1.000 | 1.247 | 3.190 | 0.493 | 0.000 | 0.472 | 0.622 |
| B04 | File-size difference (byte) | 6,400 | 264,609 | 2,074 | 2,228 | 0 | 310,168 | 25,650 |
| | File-size change rate | 1.000 | 1.247 | 3.096 | 0.489 | 0.000 | 0.464 | 0.548 |
| B05 | File-size difference (byte) | 8,013 | 332,352 | 1,588 | 2,549 | 0 | 422,366 | 33,684 |
| | File-size change rate | 1.000 | 1.248 | 2.963 | 0.490 | 0.000 | 0.467 | 0.541 |

About the comparison of the abilities of resistance of the above-mentioned five existing methods and the proposed one to text reformatting “attacks,” it is noted at first that the resistance against an attack is defined to be the fact that after the specific attack, the hidden message can still be extracted out correctly [3, 24-25]. In the proposed method, five text reformatting actions are chosen as the attacks. Text reformatting operations performed by the use of a text editing software package are often necessary to increase the readability of the XHTML codes of the EPUB file of the e-book. However, some of such operations will cause alterations to the XHTML codes and destroy the embedded message bits, and so may be regarded as attacks to the EPUB file. Five of such text reformatting operations have been identified and their attack effects to the stego-file yielded by the above-mentioned six methods have been Fig.d out in this study and listed in **Table 6**, including:

(1) eliminating excessive blank spaces in tags and words;
(2) eliminating excessive blank spaces and tabs at line ends;
(3) normalizing attribute names into small letters;
(4) indenting and reformatting tag structures; and
(5) rearranging the order of attributes.

From the table, it can be seen that the proposed method can resist all of the five types of attacks, while some of the other methods cannot. More specifically, except the method proposed by Inoue et al. [19](a), each of the other four methods has one or two weaknesses of resisting reformatting attacks.

**Table 6.** A comparison of the resistance capabilities of five methods and the proposed method to attacks by text reformatting

| Attack | Kabetta et al. [14] | Lee & Tsai [17] | Inoue et al. [19](a) | Inoue et al. [19](b) | Huang et al. [20] | Proposed method |
|---|---|---|---|---|---|---|
| Eliminating excessive blank spaces in tags and words | Yes | **No** | Yes | **No** | Yes | Yes |
| Eliminating excessive blank spaces and tabs at line ends | **No** | Yes | Yes | Yes | Yes | Yes |
| Normalizing attribute names into small letters | Yes | Yes | Yes | Yes | Yes | Yes |
| Indenting and reformatting tag structures | **No** | Yes | Yes | Yes | Yes | Yes |
| Rearranging the order of attributes | Yes | Yes | Yes | Yes | **No** | Yes |

## 6. Discussions and Conclusions

In this section, how the proposed method can be used to achieve the goal of copyright protection is discussed at first, followed by the conclusions of this study.

### 6.1 Discussions

The proposed method is a data hiding scheme for copyright protection of e-books. It is basically a text steganography method [3] suitable for embedding bit-string data into XHTML files, and so can be applied to copyright protection of e-books in the EPUB format which includes XHTML files. Specifically, as mentioned previously in Section 1 – Introduction, three common types of infringement, namely, *download*, *copy*, and *modification* in the issues of copyright protection of e-books can be resolved by the proposed method as described in the following. (1) If an EPUB file of an e-book has been protected by the proposed method with copyright information (e.g., a piece of text describing the name of the owner of the e-book) embedded into the XHTML file by the proposed data embedding algorithm (Algorithm 1), then if the e-book is *downloaded illegally from a website*, the copyright information (i.e., the owner's name) can be extracted out of the downloaded file by the proposed data extraction algorithm (Algorithm 2) to prove the copyright. And this is what "copyright protection from

*downloading*" is meant in this study. (2) If an e-book with a protected EPUB file is *copied illegally, say, from another person's computer*, then it can be proved to be an infringement by the same way as that for proving illegal downloading described previously. Or if the copy is partial, then such a kind of partial infringement can still be proved by the proposed method by embedding *repetitively* a piece of copyright-information text into the EPUB file in advance using the proposed data embedding algorithm (Algorithm 1); and the partial copy will so include one or more repetitive copyright-information texts that can be extracted out by the proposed data extraction algorithm (Algorithm 2) for copyright verification. And this is what "copyright protection from *copying*" is meant in this study. (3) If the text of an e-book with a protected EPUB file is *modified* (i.e., *attacked*) *by some specific reformatting operations* as described in Section 5.3, the proposed method can resist it by extracting the copyright information correctly using the proposed data extraction algorithm (Algorithm 2). And this is what "copyright protection from *modification*" is meant in this study.

### 6.2 Conclusions

A new method based on the technique of integer factorization for data hiding via e-books in the EPUB format for the copyright protection application has been proposed. At first, a fixed number $m$ of message bits is taken out and transformed into a decimal value based on which $k$ integer factorizations are generated. Then, an integer value $n$ is computed to be the binary logarithm of $k$, and accordingly $n$ message bits are taken out further and transformed into a decimal value for use to choose one factorization, denoted as $a \times b$, from the $k$ ones. Then, according to the values of $a$ and $b$, a combined use of the <p> and  elements in the XHTML files of the e-book is carried out to embed the $m + n$ taken message bits by including into the two elements a pre-defined class selector named according to the value of $a$ and a text paragraph with $b$ characters. The class selector is created by the use of a CSS pseudo-element.

Experiments with the EPUB files of 20 e-books as inputs have been conducted to test the proposed algorithms, and good results show the feasibility of the proposed method for data hiding via the CSS and XHTML codes in the EPUB file of the e-book. Also, comparisons of the proposed method with five existing related ones from the viewpoints of message-bit embedding capability, file-size difference and file-size change rate, and resistance to text reformatting attacks have been conducted. The results show that the proposed method outperforms the five existing methods. Finally, the security of the embedded data is protected by randomizing the message bits using a random number generator and a secret key before the message bits are embedded. The EPUB file with the embedded message yields web pages of no difference from the original ones in appearance. These experimental results and comparisons show the effectiveness of the proposed method for copyright protection of e-books. The method, which is based on the integer factorization technique, is not proposed before.

## References

[1] C. Y. Tsai, C. Y. Yang, I. C. Lin, and M. S. Hwang, "A Survey of E-book Digital Right Management," *International Journal of Network Security*, vol. 20, no.5, pp. 998-1004, 2018. Article (CrossRef Link).

[2] J. Van Tassel, *Digital rights management: Protecting and monetizing content*, Waltham, MA, USA: Focal Press, 2006. Article (CrossRef Link).

[3] I. Cox, M. Miller, J. Bloom, J. Fridrich and T. Kalker, *Digital Watermarking and Steganography*, Burlington, MA, USA: Morgan Kaufmann, 2007. Article (CrossRef Link).

[4] L. Pérez-Freire and F. Pérez-González, "Spread-spectrum watermarking security," *IEEE Transactions on Information Forensics and Security*, vol. 4, no.1, pp. 2-24, 2009. Article (CrossRef Link).

[5] P. Campisi, D. Kundur and N. Neri, "Robust digital watermarking in the ridgelet domain," *IEEE Signal Processing Letters*, vol. 11, no. 10, pp. 826-830, 2004. Article (CrossRef Link).

[6] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, pp. 354-362, 2006. Article (CrossRef Link).

[7] Wikipedia, the Free Encyclopedia, "EPUB," June, 2019. [Online] Available: https://en.wikipedia.org/wiki/EPUB.

[8] W3C, "Introduction to CSS3," May, 2019. [Online] Available: https://www.w3.org/TR/2001/WD-css3-roadmap-20010523.

[9] W3C, "XHTML Basic 1.1 - Second Edition," May, 2019. [Online] Available: https://www.w3.org/TR/xhtml-basic.

[10] Y. L. Lee and W. H. Tsai, "A new secure image transmission technique via secret-fragment-visible mosaic images by nearly reversible color transformations," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, pp. 695-703, 2014. Article (CrossRef Link).

[11] S. Bhalerao, I. A. Ansari, A. Kumar and D. K. Jain, "A reversible and multipurpose ECG data hiding technique for telemedicine applications," *Pattern Recognition Letters*, vol. 125, pp. 463-473, 2019. Article (CrossRef Link).

[12] D. C. Wu, C. Y. Hsiang and M. Y. Chen, "Steganography via MIDI files by adjusting velocities of musical note sequences with monotonically non-increasing or non-decreasing pitches," *IEEE Access*, vol. 7, pp. 154056-154075, 2019. Article (CrossRef Link).

[13] S. Das, K. Muhammad, S. Bakshi, I. Mukherjee, P. K. Sa, A. K. Sangaiah and A. Bruno, "Lip biometric template security framework using spatial steganography," *Pattern Recognition Letters*, vol. 126, pp. 102-110, 2019. Article (CrossRef Link).

[14] H. Kabetta, B. Y. Dwiandiyanta and Suyoto, "Information hiding in CSS: a secure scheme text-steganography using public key cryptosystem," *International Journal on Cryptography and Information Security*, vol. 1, pp. 13-22, 2011. Article (CrossRef Link).

[15] J. X. Lai, Y. C. Chou, C. C. Tseng and H. C. Liao, "A large payload webpage data embedding method using CSS attributes modification," *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, Pan, J. S., et al. (Eds.), Berlin, Germany: Springer, 2017, pp. 91-98. Article (CrossRef Link).

[16] D. C. Wu and H. Y. Su, "Steganography via E-Books with the EPUB Format by Rearrangements of the Contents of the CSS Files," *IEEE Access*, vol. 8, pp. 20459-20472, 2020. Article (CrossRef Link).

[17] I. S. Lee and W. H. Tsai, "Secret communication through web pages using special space codes in HTML files," *International Journal of Applied Science and Engineering*, vol. 6, pp. 141-149, 2008. Article (CrossRef Link).

[18] S. Dey, H. Al-Qaheri and S. Sanyal, "Embedding secret data in HTML web page," *Image Processing & Communications Challenges*, R. S. Choras and A. Zabtudowski, (Eds.), Warsaw, Poland: Academy Publishing House EXIT, 2009, pp. 474-481. Article (CrossRef Link).

[19] S. Inoue, K. Makino, I. Murase, O. Takizawa, T. Matsumoto and H. A, Nakagawa, "A proposal on information hiding methods using XML," in *Proc. of 1st NLP and XML Workshop*, Tokyo, Japan, 2001. Article (CrossRef Link).

[20] H. Huang, S. Zhong and X. M. Sun, “An algorithm of webpage information hiding based on attributes permutation, ” in *Proc. of International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Harbin, China, pp. 257-260, Aug. 2008.
Article (CrossRef Link).
[21] W3C, “CSS Syntax,” May 05, 2019. [Online]. Available: https://www.w3schools.com/css/css_syntax.asp.
[22] Eupor, “25 Sites To Download Free EPUB Ebook,” 03-Oct-2019. [Online]. Available: https://www.epubor.com/25-sites-to-download-free-epub-ebooks.html.
[23] Project Gutenberg, “Free eBooks - Project Gutenberg,” Oct. 13, 2019. [Online]. Available: https://www.gutenberg.org/
[24] Y. Wang, J. Liu, Y. Yang, D. Ma, and R Liu, “3D model watermarking algorithm robust to geometric attacks,” *IET Image Processing*, vol. 11 no. 10, pp. 822-832, 2017.
Article (CrossRef Link).
[25] X. Liu, S. Sun, J. Zhu, M. Jiang and D. Xie, "An Audio Watermarking Method of Resistance Statistics Attack Based on Psychoacoustic Model," in *Proc. of 2009 Fifth International Conference on Information Assurance and Security*, Xi'an, China, pp. 737-739, 2009.
Article (CrossRef Link).

**Da-Chun Wu** received the B.S. degree in Computer Science and the M.S. degree in Information Engineering from Tamkang University, Taipei, Taiwan in 1983 and 1985, respectively, and the Ph.D. degree in Computer and Information Science from National Chiao Tung University, Hsinchu, Taiwan in 1999. He joined the faculty of the Department of Information Management at Ming Chuan University, Taipei, Taiwan in 1987. From 2002 to 2018, he was with National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan. From 2010 to 2014, he was the Director of Library and Information Center of the university, and from 2015 to 2018, he was the Head of the Department of Computer and Communication Engineering. Dr. Wu is currently a Professor in the Department of Computer and Communication Engineering at National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan. His recent interests include multimedia security, image processing, and machine learning.

**Ping-Yu Hsieh** received the B.S. degree in the Department of Computer Science and Engineer Information from I-Shou University, Kaohsiung, Taiwan in 2011 and the M.S. degree in the Department of Computer and Communication Engineering from National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan in 2013. His current research interests include information hiding, networking systems, and high-frequency trading.