Binary Image Documents Authentication and Change Location Detection Based on Run Lengths

Mohsen Lesani, Mansour Jamzad

Abstract—**The need for document authentication emerges when enterprise workflows are made paperless and documents are published and claimed electronically via the internet. The current paper aims to propose an authentication and change location detection for binary images that almost all the official documents can be represented as. The previously method of adjusting global length of runs is compared with the new method with the idea of localizing data hiding and extraction. The results show that the new method outperforms the previous method in change location detection especially when more than one change is performed in the document image.**

Index Terms—**Authentication, Change Location Detection, Image Documents, Data Hiding, Binary Images**

I. Introduction

With the rapid growth of Internet, there are lots of electronic document transformations via the Internet. There is a need to protect the intellectual property rights and authenticate electronic documents. Consider publishing a patent image file with a patent number and the name of the person to whom the patent is granted. This image file will have the same legal characteristics of a paper document. Considering the fact that changing an image document is much easier than changing the paper document what happens if someone changes the patent number, patent owner name or other contents of the image by editing the file with an image processor and offer it as a legitimate document? How can one distinguish between an officially published document and a cheated one? One solution is to hide some data like trademarks, copyright signals or secret words in documents that are published and authenticate receiving documents by extracting the same data.

Binary images are images that have black or white color for each pixel. As official and legal documents are generally colorless it seems proper to consider them as binary images. Hiding data in binary images is simple as only two colors should be dealt with, on the other hand it can be hard as two colors in a binary image are extreme contrast and it is easy to be perceived by human's eyes when the colors of some pixels are changed. Therefore from this point of view it is more difficult to hide data into binary images than true color images or gray-level images.

Yu-Chee and others have devised a mathematical method for data hiding in binary images in [2], but their work has not directly considered authentication. Tseng Da-Chun Wu and others have proposed a method for data hiding and authentication of binary images based on length of runs [1]. There method truly authenticates a document but can not properly indicate the location of change in the document, especially when more than one part of the document is changed.

This paper proposes a method for not only data hiding and authentication of binary images but also indicating the location of change in a cheated document. The results of the proposed method are compared with the results of Wu method and show that more precise detection of change location is observed and reported.

The remainder of this paper is organized as follows. Section II introduces runs and length of runs in binary images. Section III describes Wu's method for data hiding and authentication. Section IV illustrates the proposed method. The results come in Section V and Section VI finishes the paper with a conclusion.

II. Runs in Binary Images

One way of storing and viewing a binary image is by using a sequence of runs. RLE (Run Length Encoding) is a compression technique that is commonly applied to binary images in fax machines in which the image is scanned by a raster order. A run or block is a continuous sequence of horizontal block of pixels with the same color. White and black runs are obviously the runs with white and black color pixels. The essential parameters of each run are its color and length. Based on the run concept, an image can be viewed as a series of alternating white and black runs starting with a white or black run. Each run-pair is composed of two adjacent runs. A run-pair can start with a white or black run based on an agreement. Fig. 1 illustrates this graphically.

The idea is that the points where white runs join black runs are coarse boundaries within a binary image. If we make slight

This paper is written in June 2005.

Mohsen Lesani is studying for MS degree in Artificial Intelligence at the Computer Engineering Department, Sharif University of Technology, Tehran, Iran (e-mail: mohsen_lesani@mehr.sharif.edu)

Dr. Mansour Jamzad is with the Computer Engineering Department at Sharif University of Technology, Tehran, Iran (e-mail: jamzad@sharif.edu).

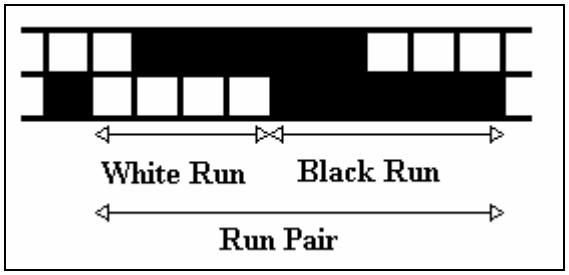

Fig. 1. A Run pair

modifications like one pixel color reversion on these locations, the change will not be easily perceived by human's eyes.

III. Data Hiding and Authentication Based on Global Length of Runs Adjustment

A. Data Hiding

The watermark that is embedded into the image is a bit stream that can come from another image like a logo image, a text, a data file, etc. This is the data that should match the data that is extracted from a document image offered to be authenticated. Run pair boundary is the place where a change in one of the two pixels with the opposite colors meeting at the boundary can be performed without much attraction of the viewer attention. If we change a pixel color on each run boundary then we are able to embed one bit per run-pair.

The image run pairs and the bit stream to be embedded are iterated cosequentially. Each bit coming from the bit stream should hide in a corresponding run pair. The odd/even characteristic of the length of the black run in a run pair is used to represent the embedding bit and an agreement is set that if the black run length is even, the embedded data is 0 and otherwise if the black run length is odd, the embedded data is 1. The color of the boundary pixel in a run-pair is changed to satisfy the condition of black run length for the embedding bit of that run pair if the black run length does not already satisfy the needed condition. Changing the length of a black run to the other parity if needed can be achieved by making the white pixel on the boundary black or making the black pixel on the boundary white. But shrinking the bigger run makes less perceivable change than shrinking the smaller one. Thus the data hiding algorithm will be as follows:

```
embeddingImage hideData(originalImage,
bitStreamToHide)
{
  embeddingImage = originalImage
  While (there are more run pairs in the original
  image)
  {
    get the next run pair satisfying the length
    condition from the embedding image.
    get the next bit to embed from the bit stream.
    if the bit to embed is zero
        if the length of the black run is odd
            Make the black run length even by
            moving the boundary to shrink the
            bigger run by one pixel.
        //else (If the length of black run is
        even)
            // Do nothing.
    else (if the bit to embed is one)
        if the length of the black run is even
            Make the black run length odd by
            moving the boundary to shrink the
            bigger run by one pixel.
      // else (If the length of black run is odd)
            // Do nothing.
  }
  return embeddingImage
}
```

The change of quality of image might be easy to perceive after an adjustment for a run with a small length. For example, if the number of pixels in a run is originally 1, it may vary from 1 to 2 doubling the length. Therefore, the method uses a controlled threshold T for filtering out the runs with too small lengths. The quality of the modified image can then be controlled this way. So the condition for a run pair to embed a bit in is:

$$\begin{aligned} &if \quad (L_B > L_W \geq T \quad or \quad L_W > L_B \geq T) \quad or \quad (L_B = L_W \quad and \quad L_B > T) \\ &\begin{cases} L'_B = L_B + \mathrm{sgn}(L_W - L_B) \times \left| (L_B \bmod 2) - d \right| \\ L'_W = L_W - \mathrm{sgn}(L_W - L_B) \times \left| (L_B \bmod 2) - d \right| \end{cases} \\ &where \quad \mathrm{sgn}(x) = \begin{cases} 1 \ if (x \geq 0) \\ -1 if (x < 0) \end{cases} \end{aligned} \qquad [1]$$

Where L_B, L_W, L'_B and L'_W are black run length before change, white run length before change, black run length after change and white run length after change.

To improve the image clarity, the run pairs that span to the next row of the image can be ignored or can be considered as being just started in the next row.

B. Data Extraction

The data extraction algorithm should extract the data hidden in an offered document image. Authentication validates the image document if the extracted data is the same as the data that has been used in data hiding before. The extraction algorithm iterates the run pairs and extracts the hidden bits from all run pairs which has had the embedding condition and has been used for embedding; in addition it extracts no bit from the run pairs which has had not the embedding condition. Assuming that the extracted bit is named *d'*, data extraction condition for a run pair is:

$$if \left(\begin{array}{l} (L_B > L_W \geq T \quad or \quad L_W > L_B \geq T) \quad or \\ (L_B = L_W \quad and \quad L_B > T) \end{array} \right) then$$

$$d' = L_B \bmod 2 \qquad [2]$$

The runs that do not satisfy the condition of threshold T are ignored as they have not been selected to embed any data. The "if" conditions in data hiding and extraction rules maintain this

ignorance. If a run pair does not satisfy the data hiding condition, then no data is embedded in it and its runs lengths do not change. As the data extraction condition expression is the same as the data hiding condition expression, no bit is also extracted from the run pair.

If a run pair passes the data hiding condition it has satisfied one of the expressions of the or connector of equation (1). For the first expression without loss of generality if $L_B > L_W \geq T$ then by shrinking the bigger run $L'_B = L_B - 1$ and $L'_W = L_W + 1$. If $L_B = L_W + 1$ then $L'_W > L'_B \geq T$. If $L_B = L_W + 2$ then $L'_B = L'_W$ and $L'_B > T$. If $L_B - L_W > 2$ then $L'_B > L'_W > T$. For the second expression again without loss of generality if the black run is chosen to shrink then $L'_W > L'_B \geq T$.

This showed that in all the cases where a run pair passes the condition for data hiding it will also pass the condition for data extraction. The data extraction algorithm would therefore be:

```
hiddenData extractData(embeddingImage)
{
  While (there are more run pairs in the embedding
  image)
  {
    get the next run pair satisfying the length
      condition from the embedding image.
    if the length of the black run is even
        hidden bit is zero.
    else // if the length of the black run is odd
        hidden bit is one.
    append the hidden bit to the extracted
      hiddenData vector
  }
  return hiddenData
}
```

One important characteristic of this method is that data extraction can be carried out without referencing the original document image. The method described can be used both for transferring some data from a party to another via an image and also for authenticating images.

The threshold imposes a tradeoff between capacity and image clarity when the method is used for transferring data and a tradeoff between more precise authentication and image clarity when used for authentication purposes because the lower the value of T parameter, the more run pairs can be used for embedding and extraction, hence the more parts of the image become sensitive to change.

When the method is used for authentication, a marked image can be constructed using the data that is extracted from the input image. Each run pair that the bit extracted from it is not equal to the corresponding bit from the previously embedded bit stream is marked with another color. The color depicts the locations in the image which are probably changed. This change location detection method is unable to precisely mark the changed areas as a run pair may be too long to specify a limited area. Also when a change in the image changes the number of run pairs, the image may be marked in almost all the run pairs starting from the first run pair that the change has started to the end of the image. This happens because adding or removing some run pairs makes the checking bit stream to move more rapidly or more slowly respectively in the areas which some run pairs are added or removed compared to the case when authentication is done for an image with no change. As the run pair iterator and bit stream iterator move cosequentially, making the checking bit stream move more rapidly or more slowly in some areas causes the bit stream iterated bits not to correspond to the run pairs which have been used to embed these bits in the data hiding algorithm and hence many of the run pairs are marked to the end of the image. This can provide the detection for the first change in an image while others can not be detected because the first change marks the image past all the other changes.

IV. Data Hiding, Authentication and Change Location Detection Based on Local Length of Runs Adjustment

A. Data Hiding

The problem with the previous method was with its global adjustment of length of run pairs. When the run pairs are computed and manipulated in the whole image, a change in the number of run pairs in an area of the image causes the marked image to sweep all the following run pairs. The trick is to perform the same length of run pair adjustments but locally in restricted windows.

The image is partitioned into a number of windows with predefined width and height. The previous data hiding algorithm is repeated on each window with the same bit stream to hide. The data hiding algorithm is:

```
embeddingImage newHideData(originalImage,
bitStreamToHide)
{
  While (there are more windows in the original
  image)
  {
    reset the bit stream to hide
    get the next window
    embeddingWindow = hideData(theWindow,
      bitStreamToHide)
    copy the embeddingWindow to the corresponding
      location in the embedding image
  }
  return embeddingImage
}
```

B. Authentication and Change Location Detection

For data extraction the offered image is partitioned to the same window size as the data hiding algorithm. The windows are iterated and the data hidden in any of the windows is extracted using the previous data extraction algorithm (extractData algorithm). A window is authenticated if its extracted data matches the previously used bit stream in data hiding algorithm. Any window that is not authenticated is marked as a change location. The whole offered image is authenticated if all the windows are authenticated and not marked and if not, the windows that are marked are the locations in the image that are changed. The following algorithm illustrates constructing the marked image by which the authentication is done.

```
markedImage  getMarkedImage(embeddingImage,
```

```
bitStreamUsedInHiding)
{
  markedImage = embeddingImage
  While (there are more windows in the embedding
  image)
  {
    get the next window
    windowHiddenData = extractData(theWindow)
    reset the bit stream
    if (windowHiddenData does not match the data
    of the bit stream from the beginning)
        mark the corresponding window in the
        markedImage
  }
  return markedImage
}
```

V. RESULTS

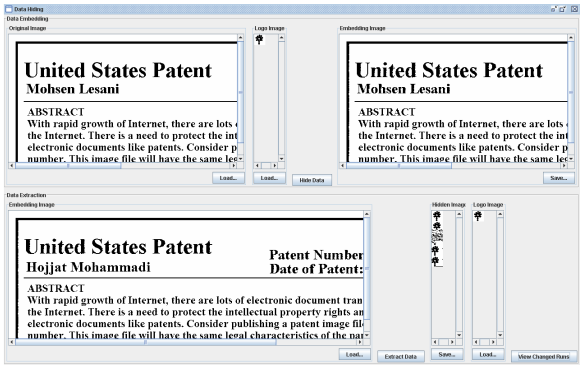

Fig. 2. Application 1

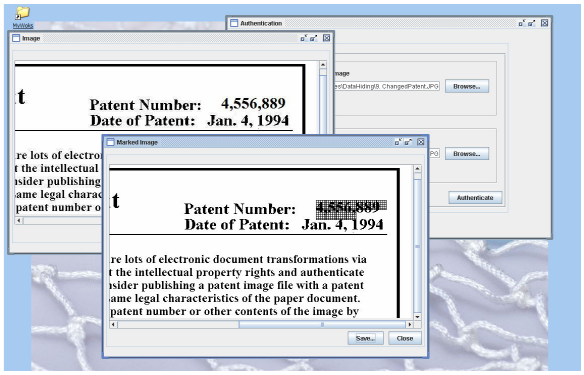

Fig. 3. Application 2

To compare the new method with the previous one, two applications are developed both in Java as shown in Fig. 2 and Fig 3. The bit stream to embed is obtained from a logo image. The clover shaped logo image used is iterated from the beginning if more bits are requested from the bit stream and the image iteration has reached the end of the image. The T parameter is set to 4 in all of the reported results. The window width and height is set to 20 and 20 pixels.

Fig. 4 shows the original patent image and Fig. 5 shows the embedding patent image that is the output of the Wu method. Fig. 6 shows the embedding patent image with its patent number changed and Fig. 7 shows its corresponding marked image. Fig. 8 shows the embedding patent image with its owner name changed and Fig. 9 shows its corresponding marked image. Fig. 10 shows the embedding patent image with both its patent number and its owner name changes and Fig. 11 shows its corresponding marked image. Fig. 12 shows the embedding patent image that is the output of the new method. Fig. 13 shows the embedding patent image with its patent number changed and Fig. 14 shows its corresponding marked image. Fig. 15 shows the embedding patent image with its owner name changed and Fig. 16 shows its corresponding marked image. Fig. 17 shows the embedding patent image with the both its patent number and its owner name changes and Fig. 18 shows its corresponding marked image.

The results show the superiority of the new method over the previous one in change location detection especially when the number of changes is more than one.

For data transferring purposes although the previous method provides more capacity, the new approach provides more robustness as the data is duplicated in different areas of the image and in the case of a change in some areas of the image the data can still be extracted from other areas.

VI. CONCLUSION AND FUTURE WORK

For authenticating image documents a new method is proposed specialized for change location detection. The results show that the new method works well at marking the changed areas. This is gained by using a local data hiding and extraction scheme.

The number of pixels that their color swap in each window can be more that one when adjusting length of run pairs is used. As the number of pixel color changes inversely influences the image quality and user satisfaction, it is desired to change fewer pixels in each window. The better approach may be to make the number of black pixels in each window even for image documents and to reject any offered image that any one of its windows black pixel counts are odd in authentication algorithm. The number of pixel color changes in each window is at most one this way. The selection of the pixel to change in each window is important; it that can be decided by morphological operations to minimize the attention of user that will be manipulated in future works.

REFERENCES

[1] D. Wu, M. Hsu and J. Jheng, “Data hiding and authentication techniques for 2-color digital documents based on adjusting lengths of runs”, 16th IPPR Conference on Computer Vision, Graphics and Image Processing (CVGIP 2003), pp. 118-822, Kinmen, ROC, 2003/8/17~19

[2] Y. Tsengy and H. Panz, "Secure and Invisible Data Hiding in 2-Color Images", INFOCOM 2001 IEEE Proceedings., Volume: 2, pp. 887-896, 2001

[3] W. Luo, “Object-Related Illustration Watermarks in Cartoon Images”, Masters Thesis, Department of Simulation and Graphics, Otto-von-Guericke University Magdeburg, Germany, February 2004.

[4] R. C. Gonzalez and R. E. Woods, “Digital Image Processing”, ISBN 0-201-18075-8, Prentice-Hall Inc., 2002

[5] R. Jain, R. Kasturi and B. G. Schunck, “Machine Vision”, McGraw-Hill, Inc., 1995

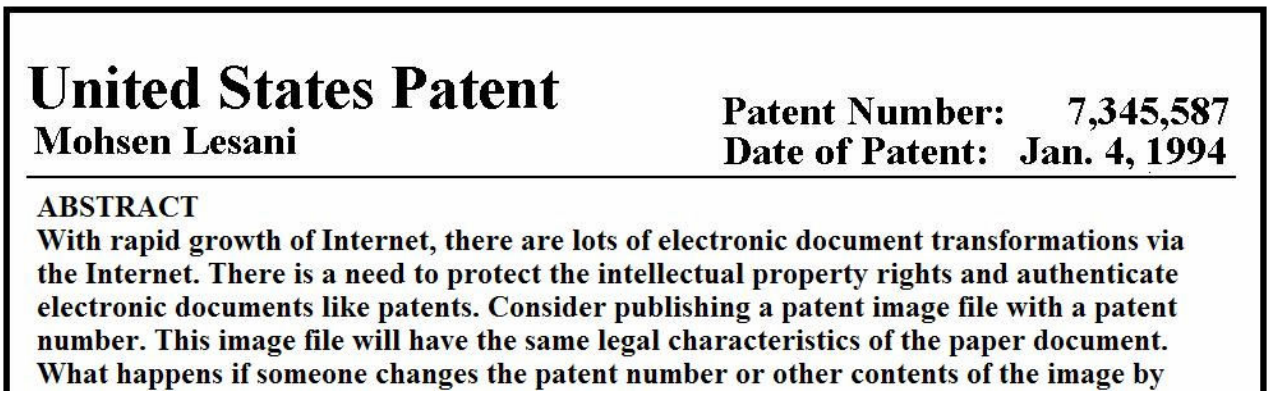

Fig. 4. Original Patent Image

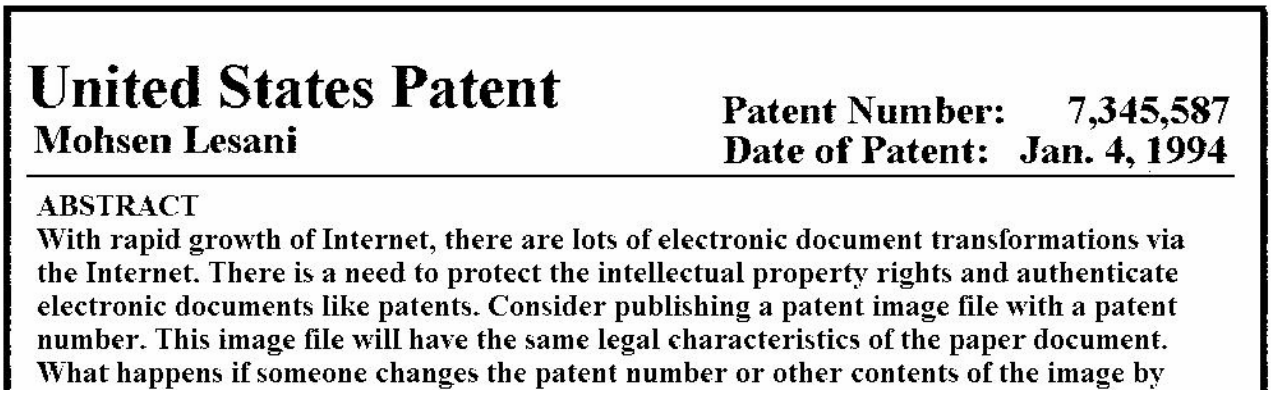

Fig. 5. Embedding Patent Image (Wu method)

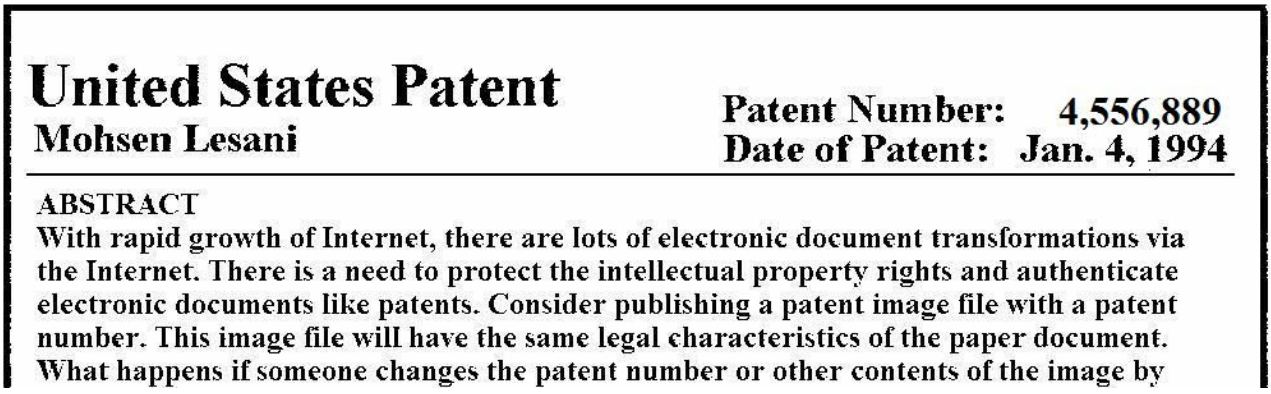

Fig. 6. Embedding Patent Image with the Patent Number Changed (Wu method)

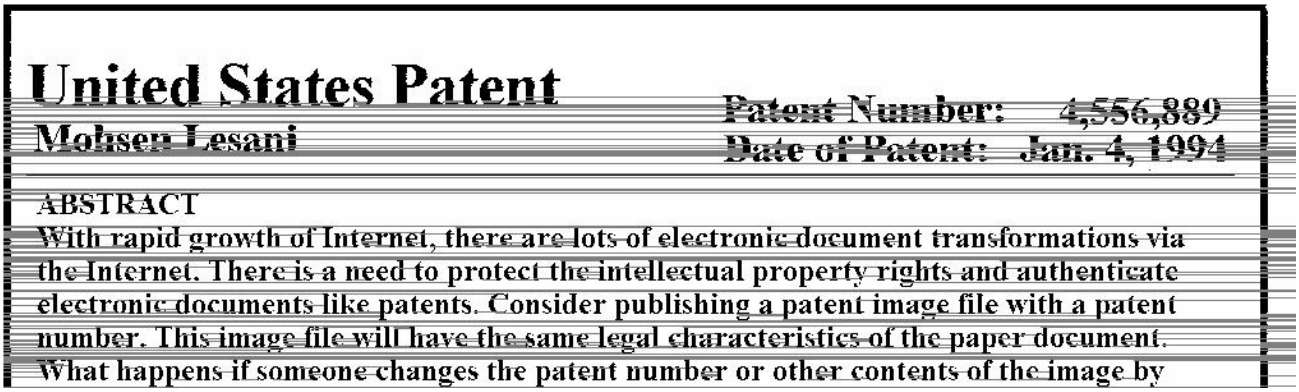

Fig. 7. Marked Image of Fig. 6 Image (Wu method)

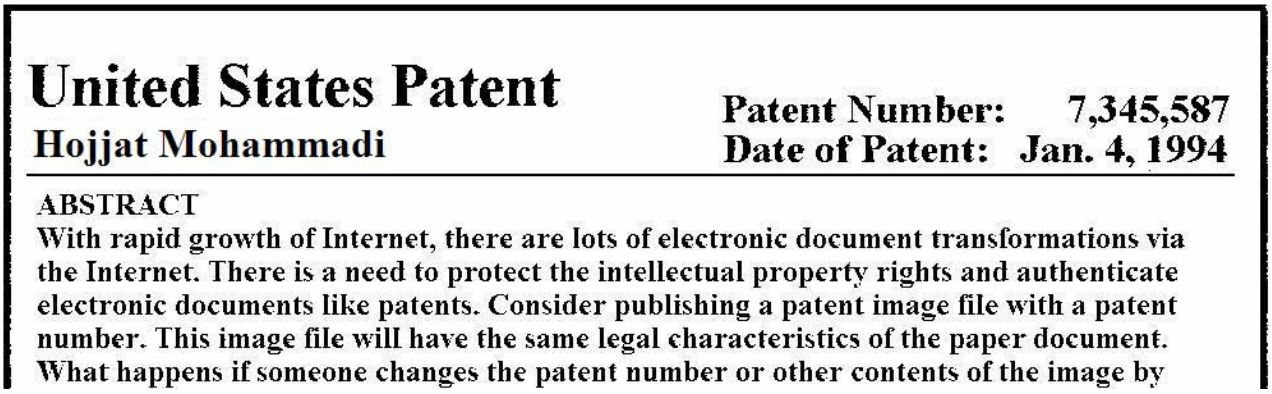

Fig. 8. Embedding Patent Image with the Owner Name Changed (Wu method)

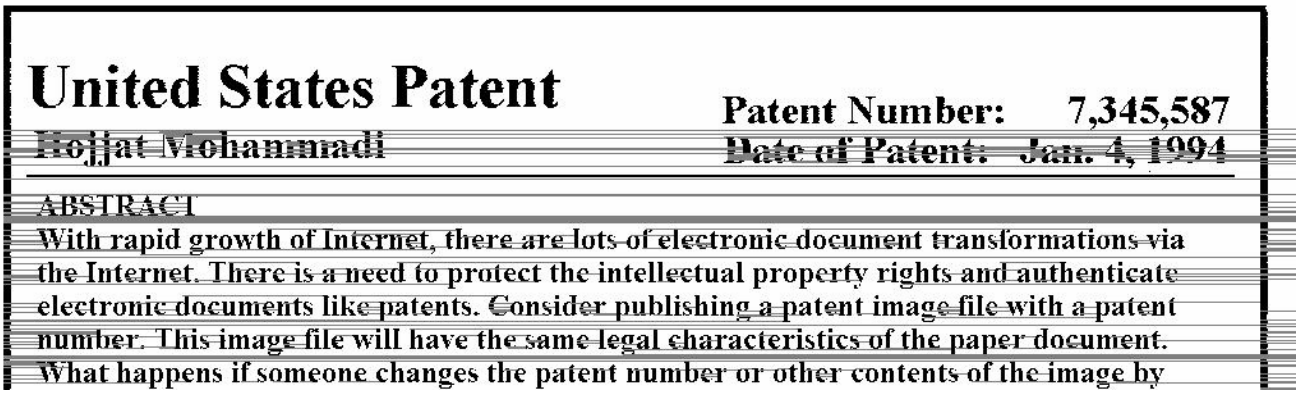

Fig. 9. Marked Image of Fig. 8 Image (Wu method)

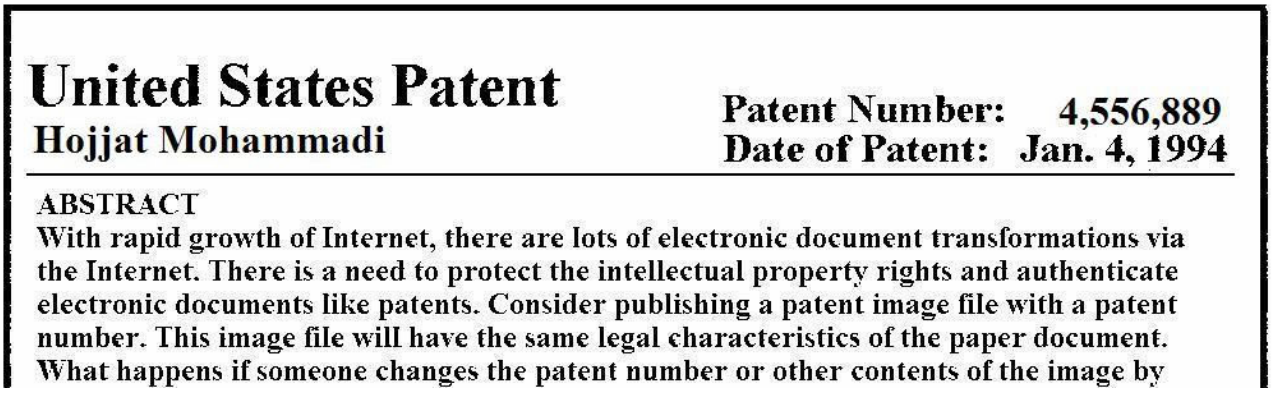

Fig. 10. Embedding Patent Image with the Patent Number and Owner Name Changed (Wu method)

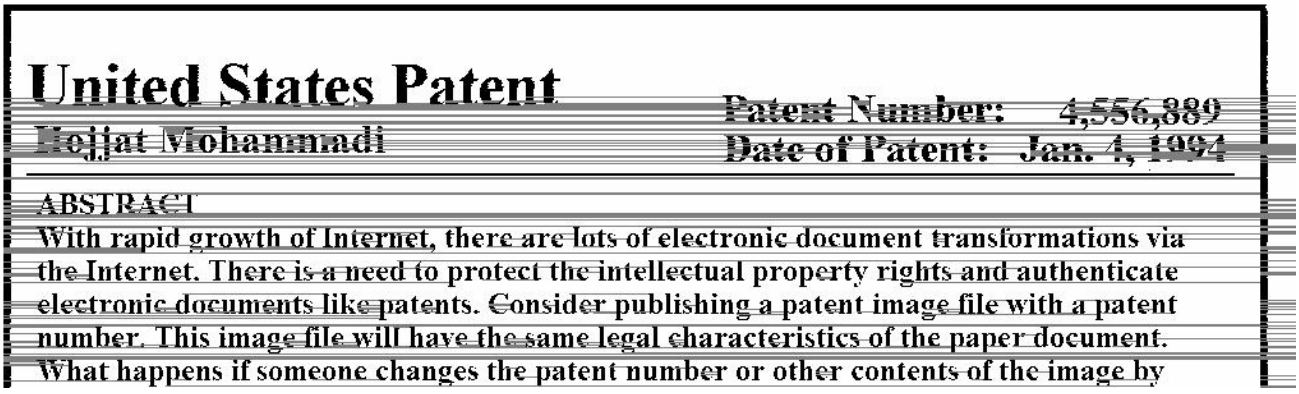

Fig. 11. Marked Image of Fig. 10 Image (Wu method)

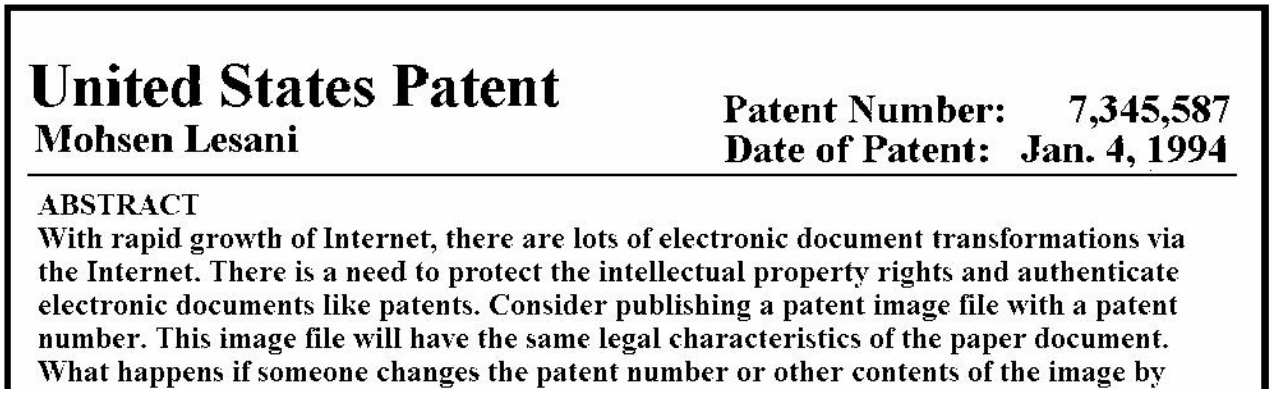

Fig. 12. Embedding Patent Image (new method)

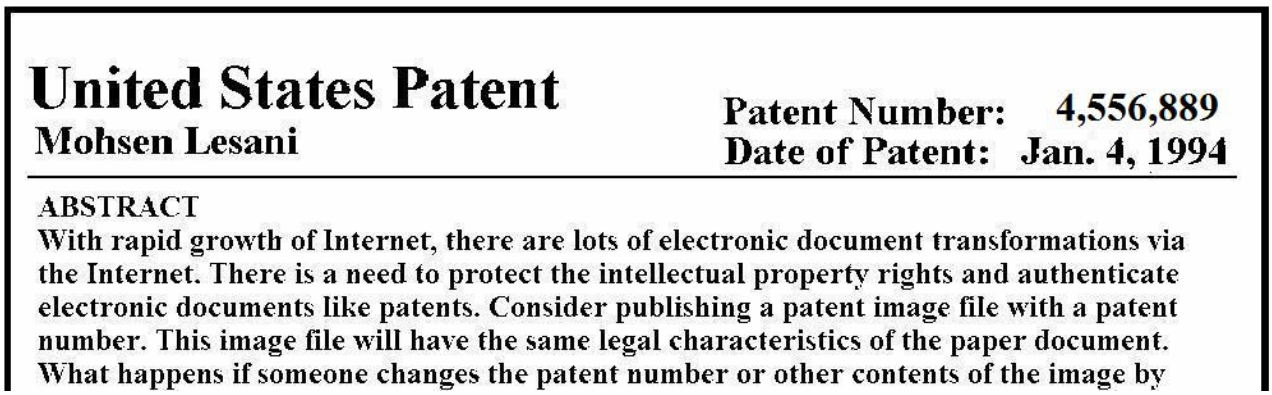

Fig. 13. Embedding Patent Image with the Patent Number Changed (new method)

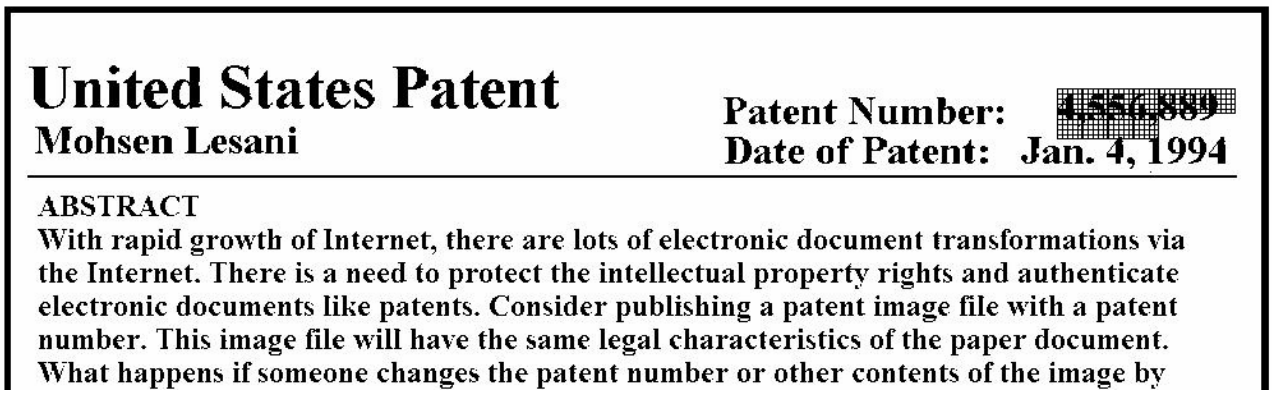

Fig. 14. Marked Image of Fig. 12 Image (new method)

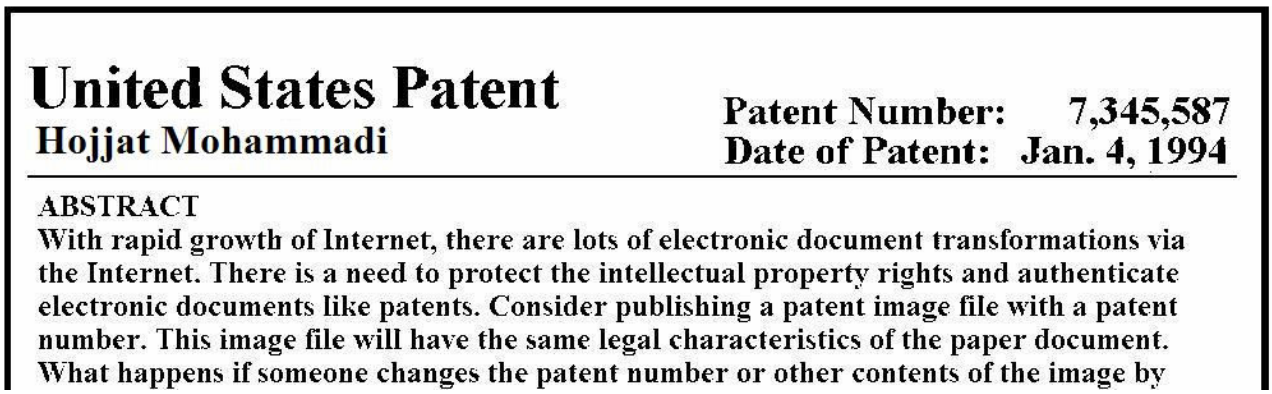

Fig. 15. Embedding Patent Image with the Owner Name Changed (new method)

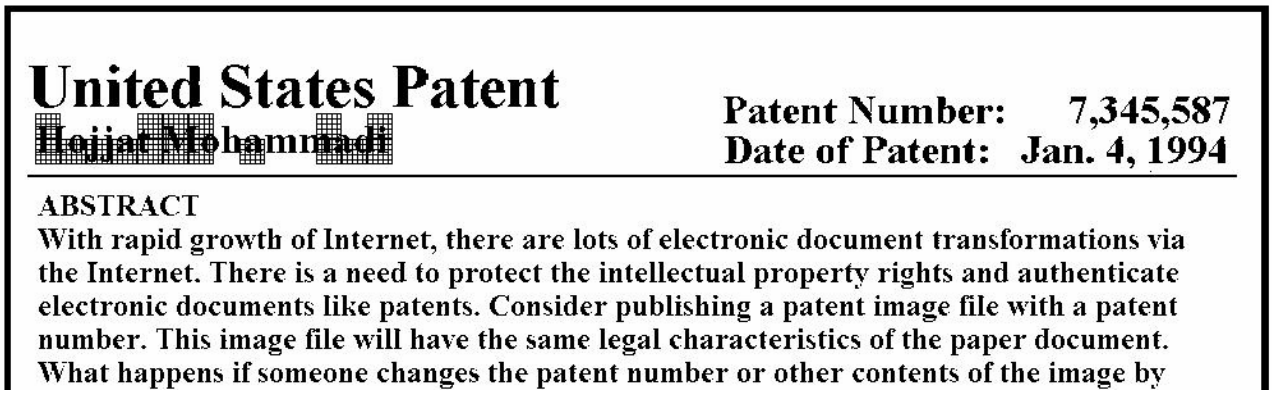

Fig. 16. Marked Image of Fig. 15 Image (new method)

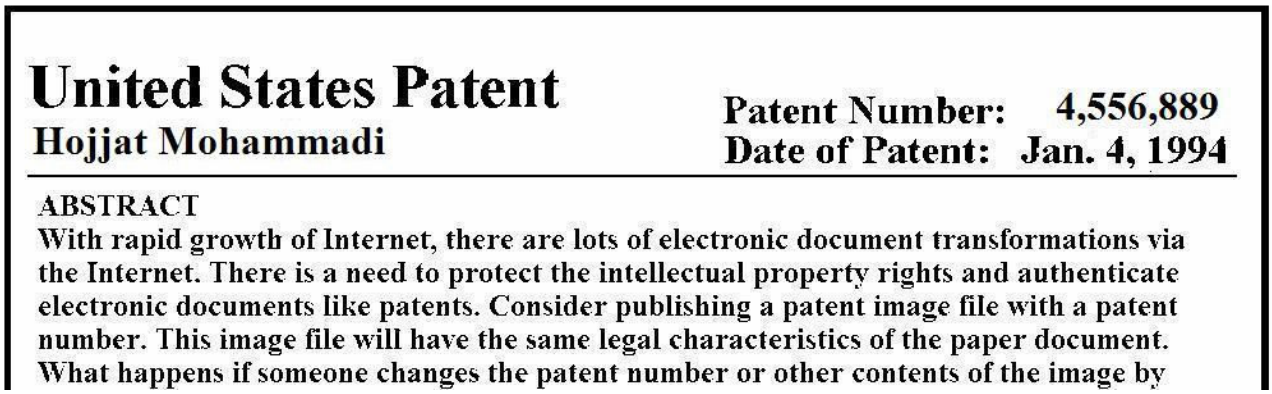

Fig. 17. Embedding Patent Image with the Patent Number and Patent Owner Changed (new method)

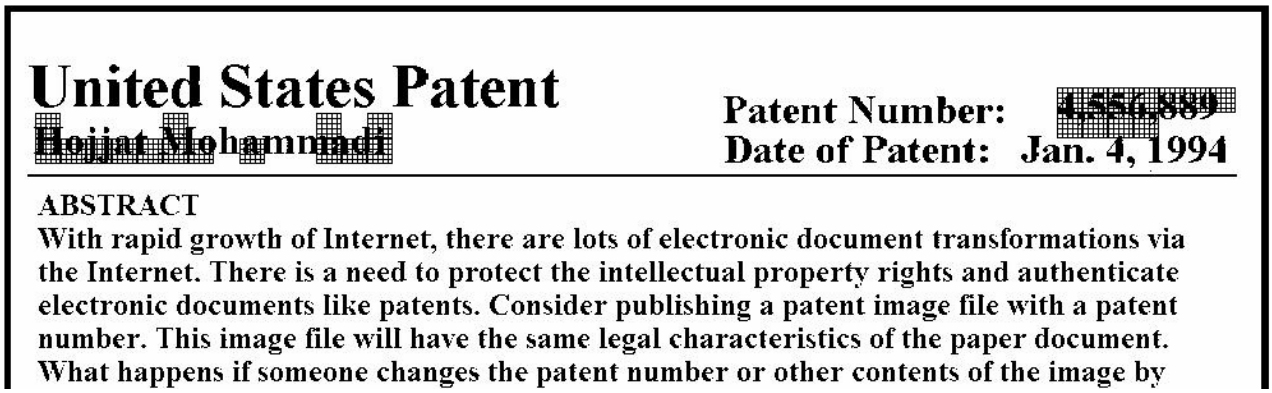

Fig. 18. Marked Image of Fig. 17 Image (new method)