# WATERMARKING JBIG2 TEXT REGION FOR IMAGE AUTHENTICATION

*Sergio Vicente D. Pamboukian[1], Hae Yong Kim[2] and Ricardo L. de Queiroz[3]*

[1]Universidade Presbiteriana Mackenzie, Brazil. E-mail: sergiop@mackenzie.com.br
[2]Universidade de São Paulo, Escola Politécnica, Brazil. E-mail: hae@lps.usp.br
[3]Universidade de Brasília, Brazil. E-mail: queiroz@ieee.org

## ABSTRACT

An authentication watermarking technique (AWT) inserts hidden data into an image in order to detect any accidental or malicious alteration to the image. AWT usually computes, using a secret-key, the message authentication code (MAC) of the image, and somehow inserts MAC into the image itself. The authentication verification is performed using the same secret-key used in the insertion. JBIG2 is an international standard for lossy and lossless binary image compression. It decomposes the image into several regions (text, halftone, line-art) and encodes each one using the most appropriate method. This paper proposes a novel data-hiding technique to embed the data into the text region of a JBIG2 file. We, then, use it to create an AWT for a JBIG2-encoded image. The embedded data can be extracted from either the JBIG2 file itself or the binary image obtained by decoding the JBIG2 file.

## 1. INTRODUCTION

In modern digital watermarking, a data-hiding scheme means a technique to embed a sequence of bits in a host image with small visual deterioration and the means to extract it afterwards. A watermarking technique makes use of a data-hiding technique to insert some information in the host image, in order to make an assertion about the image in the future. Watermarking techniques can be classified as either "robust" or "fragile." Robust watermarks are designed to resist common image-manipulation procedures (rotation, scaling, cropping, etc.) and are habitually used for the copyright verification and fingerprinting. Alternatively, fragile watermarks (or authentication watermarks) can be easily removed. However, watermarks for checking image integrity can be fragile because if the watermark is removed, the watermark detection algorithm will correctly report the corruption of the image.

In the literature, there are many authentication watermarking techniques (AWTs) for continuous-tone images [1]-[4]. Also there are many data-hiding techniques for binary images [5]-[8]. However, only very recently secure AWTs for binary images have been proposed [9]-[11]. To the best of our knowledge, no AWT for JBIG2-encoded binary images (possibly lossy-compressed) has ever been proposed. We mean by "secure AWT" a scheme that has two properties: (1) it must detect *any* visually significant image alteration (both accidental and malicious); (2) its security must not lie on the secrecy of the algorithm but on the secrecy of the key. Hence, a secure AWT usually relies upon cryptography.

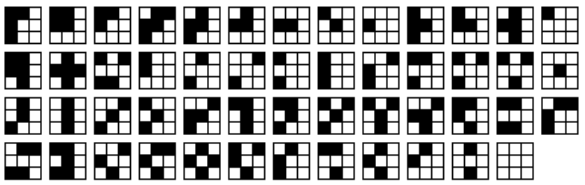

**Fig. 1:** A template ranking with all possible 3×3 templates (mirrors, rotations and reverses of each pattern have the same score).

This paper first proposes a data-hiding technique to embed data into the text region of a JBIG2 file. The embedded data can be extracted from either the JBIG2 file itself or from the binary image obtained after decoding the JBIG2 file. This technique can be used to embed data in both lossy and lossless JBIG2 files. Then, we use the proposed data-hiding technique to create a cryptography-based secure AWT for JBIG2-encoded binary images (both lossy and lossless). This AWT can be used to protect any JBIG2 file that has a text region large enough to bear the message authentication code (MAC).

The creation and the implementation of secure AWTs for JBIG2 seems to be an important practical problem. Scanned documents are largely binary images which may be protected against fraudulent alterations. Besides, binary document images must be saved in a lossy-compressed format in order to save storage space.

## 2. DATA HIDING BY TEMPLATE RANKING

This section describes briefly the data hiding by template ranking (DHTR), a technique used to embed information in binary images [5], [12], [13]. We will make use of DHTR to create our AWT for JBIG2 images:

**1)** Divide the image into small blocks (e.g., 8×8).

This work was supported in part by FAPESP under grant 2003/13752-9 and by CNPq under grants 305065/2003-3 and 475155/2004-1.

0-7803-9134-9/05/$20.00 ©2005 IEEE

**2)** Analyze the neighborhood (usually 3×3) of each pixel to rate its visual significance. Fig. 1 depicts an example of template ranking with all possible templates, ranked in increasing visual significance from left to right, top to bottom. Mirrors, rotations and reverses of each pattern have the same score. There seems to be no single best ranking, because the most visually pleasant ranking depends on the nature of the image to be marked. Other possible rankings can be found in [13], [14].

**3)** Insert one bit in each block by forcing it to have even or odd number of black pixels, to insert bits 0 or 1, respectively. If the block already has the desired parity, no change is made. Otherwise, the pixel with the lowest visual significance is flipped.

**4)** As different blocks may have different quantities of low-visibility pixels, it is suggested to shuffle the image before embedding the data.

## 3. JBIG2 FORMAT

JBIG2 standard [15] defines a compression method for bi-level images (usually black and white). It was explicitly prepared for lossy, lossless, and lossy-to-lossless image compression [16]. The JBIG2 standard does not explicitly define a standard encoder. It only defines the requirements of a compliant bitstream. A JBIG2-encoded image is decomposed into several regions (text, halftone, line-art). Each region is encoded using the most appropriate method. A JBIG2 text region has two kinds of segments:

• *Symbol dictionary segment* – contains bitmaps of the characters present in the text region.

• *Text region segment* – describes locations of characters within the text region, with references to the symbol dictionary.

There are a few ways to create a symbol dictionary [17, 18]. Many instances of a character can refer to the same symbol in the dictionary, increasing the compression rate. In lossy compression, similar instances of a symbol can refer to the same symbol in the dictionary. This happens, for example, in scanned documents where several instances of the same letter may slightly differ. If these letters refer to a unique symbol in the dictionary, the image quality decreases but the compression rate increases.

## 4. DATA-HIDING TECHNIQUES FOR JBIG2

We developed two data-hiding techniques for JBIG2.

### 4.1 Data-Hiding by Coordinate Changing for JBIG2

The first technique, named DHCCJ (data-hiding by coordinate changing for JBIG2), embeds data in the *text region segment*. Each reference in this segment is composed basically by a number that identifies the symbol in the dictionary and the position ($x$ and $y$ coordinates) of the symbol in the text region. One bit can be inserted in each reference by forcing one of the coordinates to be even or odd. Fig. 2 shows an example of an 81×81 dpi image marked using this method. The watermark was inserted in the $y$ coordinate of the symbols. Symbol displacements are clearly visible, harming the visual quality. Only images with 600 or higher dpi have presented acceptable visual quality. As the typical resolution of a scanned binary document is 300 dpi, we decided to develop another method for low-resolution images.

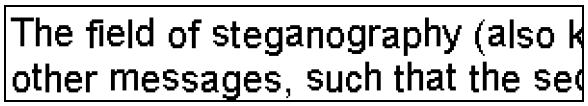


**Fig. 2:** Image marked with DHCCJ.

### 4.2 Data-Hiding by Template Ranking for JBIG2

The second technique, named DHTRJ (data-hiding by template ranking for JBIG2), embeds data in the *symbol dictionary segment*. Images watermarked by DHTRJ have good visual quality even in resolution as low as 81×81 dpi (Fig. 4). DHTRJ selects pseudo-randomly (using some seed) an appropriate number of symbols of the text region to bear the data. The quantity of the selected symbols must be enough to hold all the data to be embedded. As each symbol of the dictionary can be referred by several instances in the text region, an alteration of a dictionary symbol will have its effect multiplied, harming the visual quality of the watermarked image. The solution is to duplicate the symbols that will receive the data in the *symbol dictionary segment* and modify the *text region segment* so that only one instance of the symbol (the one selected pseudo-randomly) refers to the symbol to bear data (all others instances continue referring to the original symbol). The data-bearing symbols (DBSs) can be placed at the end of the *symbol dictionary segment* or in a new segment especially created to store them.

One bit can be inserted in each DBS by forcing it to have even or odd number of black pixels. However, if only one bit is inserted in each DBS, the dictionary size will grow significantly, affecting the power of the JBIG2 compression. The solution is to store many bits in each DBS, using a technique similar to DHTR:

**1)** Shuffle pseudo-randomly, using some seed, the set of all pixels of all DBSs.

**2)** Divide the set of shuffled pixels of DBSs into small blocks (e.g., each block with 64 pixels).

**3)** Analyze the neighborhood (usually 3×3) of each pixel to rate its visual significance.

**4)** Insert one bit in each block by forcing it to have even or odd number of black pixels.

The shuffling of the step (1) makes all blocks to have roughly the same number of low-visibility pixels. Without this shuffling, the data-hiding algorithm has to distinguish small symbols that cannot bear many bits (like point,

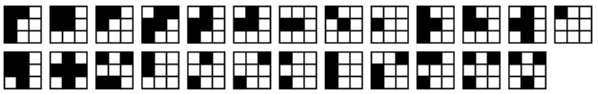

**Fig. 3:** Set of 3×3 templates that do not disconnect symbols.

comma, etc.) from large symbols. The shuffling also offers additional protection against parity attacks (to be described in section 6). We describe below some cares to be taken in the DHTRJ implementation:

**1)** The sequence of symbols within the text region must be determined using a specific method (for example, the "raster" sequence). The data insertion must not modify this order; otherwise the hidden data cannot be extracted correctly. Using the raster order, the position of the uppermost row of each symbol should remain unchanged.

**2)** The "template ranking" previously described can determine the pixel with the lowest visual significance. However, in DHTRJ, the original template set must be modified in order to eliminate templates that may cause a disconnection of the symbol. The disconnection provokes an alteration in the sequence of symbols. The set of templates that do not disconnect symbols is depicted in Fig. 3.

We have supposed a JBIG2 file with only one text region with one symbol dictionary. The adaptation of DHTRJ for JBIG2 files composed by more than one text region or symbol dictionary is straightforward. DHTRJ can be applied to both lossless and lossy JBIG2 files, because the data stored in dictionary symbols are always lossless-encoded.

## 5. THE PROPOSED AWT

DHTRJ can be converted into a secure AWT, named AWTRJ (authentication watermarking by template ranking for JBIG2). AWTRJ can authenticate any JBIG2 binary image with a text region large enough to bear the message authentication code (MAC). Usually, a 128 bits long MAC is considered secure. Using AWTRJ, only the owner of the secret-key can insert the valid watermark. Then, any visually significant alteration of the watermarked image can be detected.

AWTRJ selects pseudo-randomly (using the secret-key as the seed) a number of symbols of the text region to bear the MAC. The selected symbols are removed from the image and the resulting image (that includes not only the text region but also halftone and line-art regions) and the secret-key are used to compute the image MAC. Then, the MAC is inserted into DBSs by DHTRJ, using the secret-key as the seed of the pseudo-random shuffling of the pixels of DBSs. Then, the references to DBSs in the text region are restored.

In the watermark verification, the same pseudo-random number generator detects the marked symbols. The MAC $S$ is extracted from these symbols. After that, the marked symbols are removed from the watermarked

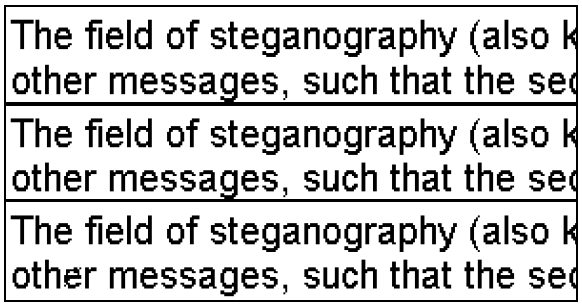


**Fig. 4. (a)** (top) part of the original image; **(b)** (center) image marked with 1 bit/symbol (in average); **(c)** (bottom) image marked with 12 bits/symbol (in average).

image and the check MAC $C$ is computed using the secret-key. If the extracted MAC $S$ is equal to the check MAC $C$, the authentication is verified. Otherwise, the image was modified.

## 6. PARITY ATTACK

AWTRJ can detect any alteration made to the portion of the image used to calculate the MAC, even if one single pixel is changed. The probability of not detecting a modification in this portion of the image is only $2^{-n}$ ($n$ is the length of MAC), which can be neglected. Unfortunately, some alterations of the data-bearing symbols (DBSs) cannot be detected. Suppose, for example, that only one bit is inserted in each symbol. In this case, the DBSs can be identified easily (e.g., DBS "a" is slightly different from the standard "a"). Any DBS can be modified, without being detected by the watermark verification, as long as the parity of black pixels does not change. A DBS "a" can be transformed into any other letter without being noticed if its parity does not change. This adulteration technique is a "parity attack" [10], [11].

Fortunately, the pseudo-random shuffling of all pixels of DBSs using a secret-key as seed reduces the possibility of parity attack, since the hacker will not know how DBSs are divided into blocks.

The technique described in [10], [11] can reduce even more the possibility of a parity attack. After inserting one bit of the MAC in a block, a new MAC must be computed, feeding the MAC computing function with the previous MAC and the modified symbol. In this way, the last DBS ($n$-th DBS) can still suffer a parity attack without being detected. However, if the DBS $n-1$ is modified maintaining its parity, this modification will be detected with 50% of chance. If the first DBS is changed (maintaining its parity), there is a probability of $1-2^{-(n-1)}$ of detecting this change.

## 7. EXPERIMENTAL RESULTS

The psycho-visual impact analysis of images watermarked by DHTRJ is beyond the scope of this paper. However, we can state that several scanned and software-generated

binary images at different resolutions (81 to 600 dpi) were watermarked by DHTRJ, resulting in images with pleasant visual quality.

The original image depicted in Fig. 4(a) has 942×306 pixels and 1025 symbol instances. It is marked with a MAC 128-bits long, stored in 128 DBSs, yielding image 4(b), which has good visual quality. Storing the MAC in 11 DBSs, the visual degradation in the marked image 4(c) is more noticeable, as in the letter "e" of the word "other".

## 8. CONCLUSIONS

This paper has proposed a new data-hiding technique to embed data into the text region of a JBIG2 binary image. The data can be inserted in both lossy and lossless JBIG2 files and can be extracted from either the JBIG2 file itself or from the decoded binary image. We used the proposed data-hiding technique to create a new cryptographically secure authentication watermarking technique for JBIG2 files.

## 9. REFERENCES


[1] M. M. Yeung and F. Mintzer, "An Invisible Watermarking Technique for Image Verification," *IEEE Int. Conf. Image Processing*, 1997, vol. 1, pp. 680-683.

[2] P. W. Wong, "A Public Key Watermark for Image Verification and Authentication," *IEEE Int. Conf. Image Processing*, 1998, vol. 1, pp. 455-459, (MA11.07).

[3] P. S. L. M. Barreto, H. Y. Kim and V. Rijmen, "Toward a Secure Public-Key Blockwise Fragile Authentication Watermarking," *IEEE Proc. Vision, Image and Signal Processing*, vol. 149, no. 2, pp. 57-62, 2002.

[4] M. U. Celik, G. Sharma, E. Saber and A. M. Tekalp, "Hierarchical Watermarking for Secure Image Authentication with Localization," *IEEE Trans. Image Processing*, vol. 11, no. 6, pp. 585-595, 2002.

[5] M. Wu, E. Tang and B. Liu, "Data Hiding in Digital Binary Image," *IEEE Int. Conf. Multimedia and Expo*, ICME'00, New York, USA, 2000.

[6] M. S. Fu and O. C. Au, "Data Hiding Watermarking for Halftone Images," *IEEE Trans. Image Processing*, vol. 11, no. 4, pp. 477-484, 2002.

[7] Y.-C. Tseng, Y.-Y. Chen and H.-K. Pan, "A Secure Data Hiding Scheme for Binary Images," *IEEE Trans. on Communications*, vol. 50, no. 8, Aug. 2002, pp. 1227-1231.

[8] S. C. Pei and J. M. Guo, "Hybrid Pixel-Based Data Hiding and Block-Based Watermarking for Error-Diffused Halftone Images," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 867-884, 2003.

[9] H. Y. Kim, and A. Afif, "Secure Authentication Watermarking for Halftone and Binary Images," *Int. J. Imaging Systems and Technology*, vol. 14, no. 4, pp. 147-152, 2004.

[10] H. Y. Kim and R. L. Queiroz, "A Public-Key Authentication Watermarking for Binary Images," in *Proc. IEEE Int. Conf. on Image Processing*, (Singapore), pp. 3459-3462, 2004.

[11] H. Y. Kim and R. L. de Queiroz, "Alteration-Locating Authentication Watermarking for Binary Images," *Int. Workshop on Digital Watermarking 2004,* Lecture Notes in Computer Science 3304, pp. 125-136, 2004.

[12] R. de Queiroz and P. Fleckenstein, "Object Modification for Data Embedding through Template Ranking," *Xerox Invention Proposal*, 1999.

[13] M. Wu, and B. Liu, "Data Hiding in Binary Image for Authentication and Annotation," *IEEE Trans. on Multimedia*, vol. 6, no. 4, pp. 528-538, 2004.

[14] Y. Fujii, K. Nakano, I. Echizen, H. Yoshiura and S. Tezuka, "A Method of Maintaining the Image Quality of Digital Watermarking for Binary Images Using Local Measures," *IPSJ Journal (Information Processing Society of Japan)*, vol. 44, no. 08, pp. 1872-1883, 2003 (in Japanese).

[15] Final Committee Draft for ISO/IEC International Standard 14492, available at site:
http://www.jpeg.org/jbig/jbigpt2.html, 1999.

[16] P.G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W.J. Rucklidge, "The Emerging JBIG2 Standard," *IEEE Trans. Circ. Syst. Video Tech.*, vol. 8, no. 7, pp. 838-848, 1998.

[17] Yan Ye, D. Schilling, P. Cosman and Hyung Hwa Ko, "Symbol Dictionary Design for the JBIG2 Standard", in *Proc. Data Compression Conference*, pp. 33-42, 2000.

[18] Yan Ye and P. Cosman, "Dictionary Design for Text Image Compression with JBIG2", *IEEE Trans. Image Processing*, vol. 10, no. 6, pp. 818-828, June 2001.