New Public-Key Authentication Watermarking for JBIG2 Resistant to Parity Attacks

Sergio Vicente Denser Pamboukian[1] and Hae Yong Kim[2]

[1] Universidade Presbiteriana Mackenzie, São Paulo, Brazil,
sergiop@mackenzie.com.br, http://meusite.mackenzie.com.br/sergio
[2] Escola Politécnica, Universidade de São Paulo, Brazil,
hae@lps.usp.br, http://www.lps.usp.br/~hae

Abstract. An authentication watermark is a hidden data inserted into an image that allows detecting any alteration made in the image. AWTs (Authentication Watermarking Techniques) normally make use of secret- or public-key cryptographic cipher to compute the authentication signature of the image, and inserts it into the image itself. Many previous public-key AWTs for uncompressed binary images can be attacked by an image adulterating technique named "parity attack." JBIG2 is an international standard for compressing bi-level images (both lossy and lossless). The creation of secure AWTs for compressed binary images is an important practical problem. However, it seems that no AWT for JBIG2 resistant to parity attacks has ever been proposed. This paper proposes a new data-hiding method to embed information in the text region of JBIG2 files. Then, we use this technique to design a new AWT for JBIG2-encoded images resistant to parity attacks. Both the secret- and public-key versions of the proposed AWT are completely immune against parity attacks. Moreover, watermarked images are visually pleasant, without visible salt and pepper noise. Image authenticity verification can be performed in either JBIG2 file itself or in the binary image obtained by decoding the JBIG2 file.

1 Introduction

Steganography (also known as data/information hiding) is the study of techniques used to hide secret information inside another kind of information, without loss of quality of the host information. For example, a sequence of bits can be embedded inside an image by modifying some of its pixels, without perceptible degradation in image quality. The steganography is not concerned about the usefulness of the hidden information or the facility of removing it. A digital watermark makes use of data hiding techniques to insert, into a digital data, a signal that can be extracted later to make an assertion about the host data. Digital watermarks are usually classified as either "robust" or "fragile," depending on the difficulty of removal.

Robust watermarks cannot be easily removed and are designed to resist common image-manipulation procedures (rotation, scaling, cropping, lossy compression, printing/scanning, etc.) This kind of watermark is normally used for copyright verification and fingerprinting.

On the other hand, fragile watermarks (or authentication watermarks) are easily corrupted by any image processing procedure. However, watermarks for checking image integrity and authenticity can be fragile because if the watermark is removed, the watermark detection algorithm will correctly report the corruption of the image.

Only recently, some secure authentication-watermarking techniques (AWTs) for uncompressed binary images have been proposed [1, 2, 3]. We mean by "secure AWT" a scheme that has two properties: (1) it must detect *any* image alteration (both accidental and malicious); (2) its security must not lie on the secrecy of the algorithm but only on the secrecy of the key. Hence, a secure AWT usually relies upon cryptography.

Many previous AWTs for uncompressed binary images can be assaulted by an image adulterating technique named "parity attack" [2, 3]. For secret-key AWTs, some general methods for preventing parity attacks have been identified [3]. However, for public-key AWTs, no general parity attack preventing method has been discovered. Only very recently, one of the authors of this paper has proposed a particular AWT for uncompressed binary images completely immune against parity attacks, named AWTC (Authentication Watermarking by Template ranking with symmetrical Central pixels) [4].

JBIG2 is an international standard for compressing bi-level images (both lossy and lossless) [5, 6]. In this standard, the image is decomposed in several regions (text, halftone and line-art) and each region is compressed using the most appropriate method. The creation and implementation of secure AWTs for compressed binary images (such as 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 stored in a compressed format in order to save storage space.

Queiroz and we have very recently proposed an AWT for JBIG2-encoded images (possibly lossy-compressed), named AWTRJ (Authentication Watermarking by Template Ranking for JBIG2) [7]. Unfortunately, AWTRJ (especially its public-key version) can be assaulted by parity attacks. To the best of our knowledge, no AWT for JBIG2 resistant to parity attacks has ever been proposed.

This paper proposes a new data-hiding method, inspired by AWTC and AWTRJ, to embed information in the text region of JBIG2 files (both lossy and lossless). The embedded data can be extracted from either JBIG2 file itself or the binary image obtained by decoding the JBIG2 file. Then, we use the proposed data-hiding technique to design a new AWT for JBIG2-encoded images resistant to parity attacks. This AWT can be used to protect any JBIG2 file (both lossy and lossless) that has a text region large enough to bear the authentication signature. Both secret-key and public-key versions of this AWT are completely immune against parity attacks.

We did not apply any perceptual distortion measure to quantify the quality of watermarked images, because this analysis is beyond the scope of this paper. However, the suggested template ranking can be adapted to minimize the distortion according to a specific perceptual model.

2 JBIG2 Format

The Joint Bi-level Image Experts Group (JBIG), a "collaborative team" established in 1988, prepared JBIG2 standard. This standard defines a compression method for bi-level images (usually black and white) and was explicitly prepared for lossy, lossless, and lossy-to-lossless image compression [5, 6]. A JBIG2-encoded image is composed by several regions (text, halftone and 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 many researches on symbol dictionary design [8, 9]. Many instances of a character can refer to the same symbol in the dictionary, what increases 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 character may differ slightly. If these similar characters refer to a unique symbol in the dictionary, the image quality decreases but the compression rate increases.

3 Previous Techniques

3.1 Data Hiding in Uncompressed Binary Images

There are three basic ways of embedding a sequence of bits in uncompressed binary/halftone images:

- **Pixel-wise**: Change the values of (usually pseudo-randomly chosen) individual pixels [10, 11]. This approach is well suited for dispersed-dot halftone images. However, visible salt and pepper noise will appear when applied to other types of binary images. It can be applied to the binary image or directly to the halftone screen in its design step [12].
- **Component-wise**: Change the characteristics of pixel groups (for example, the position or the area of connected components) [13]. Unfortunately, the success of this approach depends on the type of the host image.
- **Block-wise**: Divide the host image into blocks and modify some characteristics of each block. Some works [14, 15] suggest alternating between two different weight matrices to halftone an image such that the matrix used in each block can be determined in the future by analyzing the statistical properties. Other works suggest modifying slightly the content of the block so that it hides the desired sequence of bits [16, 17].

3.2 Authentication Watermarking for Uncompressed Binary Images

Authentication watermarking techniques (AWTs) make use of data-hiding techniques and cryptography theory to check the image integrity and authenticity. In a typical cryptography-based AWT, an authentication signature (AS) is computed from the whole image and inserted into the image itself. An AS contains information about the host image content that may be checked to verify its integrity. In cryptography, an AS is called message authentication code (MAC) using a secret-key cipher or digital signature (DS) using a public/private-key cipher.

The chosen AS must be long enough to assure the security. Too small an AS does not withstand birthday attacks. Usually, a MAC with 128-bits is considered computationally secure. The best-known DS, RSA, is considered computationally secure with 1024 bits. A newer scheme, DSA, is considered computationally secure with 320 bits. A brand new scheme, BLS, is computationally secure with only 160 bits [18]. The reader is referred to introductory books on Cryptography for more details (for example, [19]).

In a secret-key AWT, the same secret-key is used in both watermark insertion and verification. In a public-key AWT, only the owner of the private-key can insert the valid watermark, and anyone can verify the image authenticity and integrity using the corresponding public-key. However, inserting the AS into the image alters the image itself, hence modifying its AS and invalidating the watermark. Typically, the image has to be somehow divided into at least two parts: a portion to maintain the image integrity and another portion to carry the AS. However, dividing the image in two parts makes possible the occurrence of a "parity attack."

3.3 Parity Attack

Many data hiding schemes for binary images can be transformed into AWTs by simply dividing the host image Z in two regions: the first region Z_1 where the AS is to be stored, and the second region Z_2 from where the AS is to be computed. This idea was used to design AWST (Authentication Watermarking by Self Toggling) for dispersed-dot halftone images [1] and AWTR (Authentication Watermarking by Template Ranking) for generic binary images [2, 3].

However, some caution must be taken when transforming a data-hiding scheme into an AWT, because although the region Z_2 is well protected (with the security assured by the cryptography theory), the region Z_1 is not. For example, let us take the component-wise data-hiding scheme that inserts one bit per connected component, forcing it to have an even or odd number of pixels. A connected component can be forced to have the desired parity by toggling one of its boundary pixels. This scheme can be transformed into an AWT by dividing the host image in regions Z_1 and Z_2, computing the AS of Z_2 and inserting it in Z_1. Yet, a malicious hacker can arbitrarily alter the region Z_1 without being noticed by the AWT, as long as all the parities of its connected components remain unaltered. For example, a character "a" in Z_1 region can be changed into an "e" (or any other character that contains only one connected component) as long as its parity remains unchanged. We refer to this as a "parity attack."

3.4 AWTC

Very recently, a new AWT for uncompressed binary images, called AWTC (Authentication Watermarking by Template ranking with symmetrical Central pixel), was proposed [4]. This technique is completely immune against parity attacks, and consequently both its secret- and public-key versions are secure. This technique can detect *any* image alteration, even a single pixel flipping. An image watermarked by AWTC does not present visible salt-and-pepper noise. We will use the ideas used to design AWTC to create an AWT for JBIG2-encoded images immune against parity attacks.

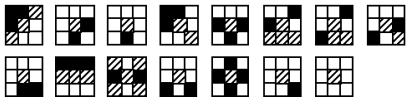


Fig. 1. A 3×3 template ranking with symmetrical central pixels in increasing visual impact order. Hatched pixels match either black or white pixels (note that all templates have hatched central pixels). The score of a given pattern is that of the matching template with the lowest impact. Mirrors, rotations and reverses of each pattern have the same score.

AWTC keeps the visual scores of flippable pixels unaltered after embedding the data. It is possible because the template ranking used assigns the same visual impact score to the patterns that differ only by the colors of their central pixels. Figure 1 depicts a 3×3 template ranking with symmetrical central pixels. Note that all patterns have hatched central pixels. To simplify the explanation, let us assume that 3×3 patterns are used, although larger patterns may be used:

1. Divide the uncompressed binary image Z to be watermarked in a sequence v of non-overlapping 3×3 pieces of image Z. The simplest of such sequence is the division of Z into regular 3×3 pieces (incomplete pieces at image borders are discarded), scanned in raster sequence (figure 2). Only the central pixels of the pieces of v can have their colors changed by the watermark insertion.
2. Sort the sequence v in increasing order using the visual scores as the primary-key and non-repeating pseudo-random numbers as the secondary-key. The secondary-key prevents from embedding the data only in the upper part of the image.
3. Clear the central pixels of the first n pieces of the sorted v, where n is the length of the AS. Compute the AS of the now-cleared image Z.
4. Embed n bits of the AS by flipping (if necessary) the central pixels of the first n pieces of the sorted v.

To extract the AS, the sequence v of non-overlapping 3×3 pieces is constructed again and sorted as in the insertion step. The result is exactly the same sequence v used in the insertion. Then, the values of the n first central pixels are the hidden data.

Why do parity attacks not apply to AWTC? Because the number of data-bearing Z_1 pixels is exactly equal to the length of the adopted AS. All image pixels (except the n pixels that will bear the n bits of the AS) are taken into account to compute the AS. Consequently, any alteration of Z_2 region can be detected because it changes the AS

of the watermarked image, and any alteration of Z_1 region can also be detected because it changes the stored AS.

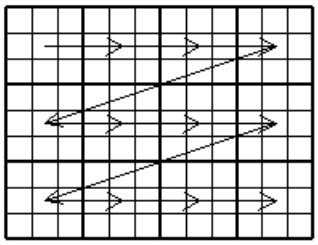


Fig. 2. A 9×12 image divided into regular 3×3 pieces and scanned in raster order.

3.5 AWTRJ

To the best of our knowledge, Queiroz and we have proposed the only AWT for JBIG2-encoded binary images (possibly lossy-compressed) [7]. This technique is named AWTRJ (Authentication Watermarking by Template Ranking for JBIG2). Only the secret-key AWTRJ has been proposed, because the public-key version can be assaulted by parity attacks. The goal of this paper is to obtain another AWT for JBIG2 immune to parity attacks and consequently secure in both secret- and public-key versions.

AWTRJ embeds the MAC in the text region of a JBIG2-encoded image, more precisely in the symbol dictionary segment. AWTRJ can authenticate any JBIG2 binary image with a text region large enough to bear the MAC. Image authenticity verification can be performed in either JBIG2 file itself or in the binary image obtained by decoding the JBIG2 file. AWTRJ consists of:

1. Selects pseudo-randomly, using the secret-key as the seed, an appropriate number of symbols of the text region to bear the data.
2. Remove the selected symbols from the image and compute the MAC of the resulting image (that includes not only the text region but also halftone and line-art regions) using the secret-key.
3. As each dictionary symbol can be referred by several instances in the text region, an alteration of a symbol will have its effect multiplied. To avoid this problem, duplicate the symbols that will bear 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 data-bearing symbol (DBS). All others instances continue referring to the original symbol.
4. Shuffle pseudo-randomly the set of all pixels of all DBSs of the symbol dictionary.
5. Divide the set of shuffled pixels of DBSs into small blocks (e.g., each block with 64 pixels).
6. Analyze the neighborhood (usually 3×3) of each shuffled pixel to rate its visual significance.

7. Insert one bit of the MAC in each block by forcing it to have even or odd number of black pixels. Only pixels that do not disconnect symbols can be flipped (figure 3).

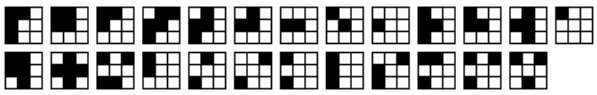


Fig. 3. Set of 3×3 templates that do not disconnect symbols.

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 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.

The secret-key AWTRJ is protected against parity attacks because it uses the secret-key as the seed of the pseudo-random selection in step 1, and uses the secret-key as the seed of the pseudo-random shuffling in step 4. Unfortunately, the same ideas cannot be applied to the public-key version, because anyone must be able to extract completely the hidden bits without knowing the private-key.

4 The Proposed Technique

We propose a new AWT for JBIG2-encoded images resistant to "parity attacks," called AWTCJ (Authentication Watermarking by Template ranking with symmetrical Central pixels for JBIG2). AWTCJ is inspired by both AWTC and AWTRJ. For the sake of clarity, we first describe the data-hiding technique DHTCJ (Data Hiding by Template ranking with symmetrical Central pixels for JBIG2), which will be transformed into AWTCJ using the idea described in subsections 3.2 and 3.3.

4.1 DHTCJ

DHTCJ data insertion algorithm is:

1. Let be given a JBIG2-encoded image *Z'* and *n* bits of data to be inserted into *Z'*. Decode the text region of *Z'*, obtaining the uncompressed binary image *Z*.
2. Divide *Z* in a sequence *v* of non-overlapping pieces of image and sort *v* as in AWTC.
3. Identify in the *text region segment* the symbols that contain the *n* first central pixels of the sorted sequence *v* and its references to the Data Bearing Symbols (DBSs) in the *symbol dictionary segment*. Note that the number of DBSs can be smaller than *n*, because each symbol can bear more than one bit of AS.
4. Verify how many times each DBS is referenced in the *text region segment*. If there's only one reference, the data will be stored in the original symbol. If there's more than one reference, the symbol must be duplicated and inserted at the end of

symbol dictionary segment. The reference to the symbol in the *text region segment* should also be modified. The data will be inserted in the duplicated symbol, instead of the original.

5. Insert *n* bits of data in the DBSs by flipping, if necessary, the *n* first central pixels of the sorted sequence *v*.
6. Verify the possibility of connection or disconnection of the DBSs. If a black pixel was transformed to white, a disconnection can occur separating the symbol in two parts (figure 4(b)). If a white pixel was transformed to black, two symbols can become united (figure 4(c)). In these cases, the DBS(s) must be eliminated from *symbol dictionary segment* and the new symbol(s) must be inserted. Also the reference(s) in the *text region segment* must be modified.

(a) Original (b) Disconnection (c) Connection

Fig. 4. A symbol may become disconnected or two symbols may become connected by the watermark insertion.

In order to simplify the implementation, we suggest using a new template ranking (figure 5) instead of the template ranking designed for AWTC (figure 1). The new set does not contain templates that can cause the connection or disconnection of symbols. Using the new template ranking, the last step of the DHTCJ insertion algorithm (that is too hard to implement) can be ignored.

Every possible 3×3 pattern has a matching template in the old template ranking (figure 1). On the contrary, there are many 3×3 patterns that do not have a matching template in the new template ranking (figure 5). This means that there may exist some small images that can hide a certain number of bits using the old template ranking (although probably some high visual impact pixels have to flipped), but that cannot hide the same number of bits using the new template ranking.

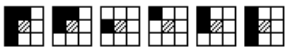


Fig. 5. Set of 3×3 template designed to be used with AWTCJ in increasing visual impact order. Only the templates that cannot cause symbol connection or disconnection are listed. Hatched pixels match either black or white pixels (note that all templates have hatched central pixels). The score of a given pattern is that of the matching template with the lowest impact. Mirrors, rotations and reverses of each pattern have the same score.

DHTCJ data extraction algorithm is straightforward:

1. Let be given a JBIG2-encoded image *Z'* with *n* bits of data to inserted by DHTCJ. Decode the text region of *Z'*, obtaining the uncompressed binary image *Z*.
2. Divide the binary image *Z* in a sequence *v* of non-overlapping pieces of image and sort *v* as in the insertion.
3. Extract the hidden data from the *n* first central pixels of the sorted sequence *v*.

4.2 AWTCJ

DHTCJ can be easily transformed in a secure AWT resistant to "parity attacks" named AWTCJ (Authentication Watermarking by Template ranking with symmetrical Central pixels for JBIG2). AWTCJ insertion algorithm is:

1. Let be given a JBIG2-encoded image *Z'*. Decode the text region of *Z'*, obtaining the uncompressed binary image *Z*.
2. Divide *Z* in a sequence *v* of non-overlapping pieces of image and sort *v* as in DHTCJ.
3. Clear the first *n* central pixels of the sorted sequence *v*, where *n* is the size of the adopted AS.
4. Using a cryptographically secure hashing function, compute the integrity-index *H* of the now-cleared image *Z*. Besides the image *Z*, all other regions of *Z'* to be protected (halftone and line-art) must be taken into account to compute *H*.
5. Encrypt the integrity-index *H* with the secret- or private-key, obtaining the AS *S*.
6. Insert *n* bits of *S* in the DBSs as explained in DHTCJ, obtaining the watermarked image.

AWTCJ verification algorithm is:

1. Let be given an AWTCJ-watermarked JBIG2-encoded image *Z'*. Decode the text region of *Z'*, obtaining the uncompressed binary image *Z*.
2. Divide *Z* in a sequence *v* of non-overlapping pieces of image and sort *v* as in the insertion.
3. Extract the AS *S* from the *n* first central pixels of the sorted sequence *v*.
4. Decrypt *S* with the secret- or public-key, obtaining the extracted integrity-index *H*.
5. Clear the first *n* central pixels of the sorted sequence *v*.

6. Compute the check integrity-index *C* of the now-cleared image *Z*, using the same hashing function used in insertion. Besides the image *Z*, all other protected regions of *Z'* must be taken into account to compute *C*.
7. If the extracted integrity-index *H* and the check integrity-index *C* are the same, the watermark is verified. Otherwise, the image was modified.

Parity attacks do not be apply to AWTCJ because the number of data-bearing pixels is exactly equal to the length of the adopted AS. All image pixels (except the pixels that will bear the bits of the AS) are used to compute the AS. In this way, any alteration of pixels used to compute the AS can be detected because it changes the AS, and any alteration of data-bearing pixels can also be detected because it changes the stored AS.

5 Experimental Results

AWTCJ was applied in several scanned and software-generated binary images at different resolutions. The resulting watermarked images have pleasant visual quality, even when a small image is watermarked.

The image depicted in figure 6 has 626×240 pixels, 93 symbols instances and was scanned at 300×300 dpi. It was watermarked by AWTCJ using a 128-bits long MAC, which was stored in 61 DBSs.

The image depicted in figure 7 has 194×74 pixels, only 56 symbol instances and was scanned at 81×81 dpi. It was watermarked with a 128-bits long MAC, which was stored in 42 DBSs.

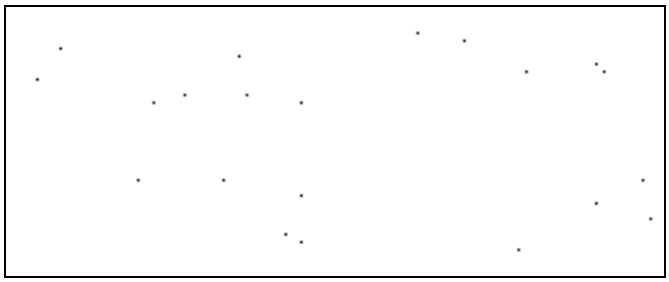
6 Conclusions

This paper has proposed a new data-hiding technique, named DHTCJ, to embed data into the text region of JBIG2-encoded images. The data can be inserted in lossy or lossless JBIG2 files and can be extracted from either the JBIG2 file itself or from the decoded binary image. Then, we have used DHTCJ to create a new cryptographically secure authentication watermarking technique for JBIG2 files, resistant to "parity attacks," named AWTCJ. In this method, the authentication signature of the whole image is inserted into the text region of the JBIG2 file. AWTCJ can detect *any* alteration made in the image, even a single pixel flipping. The watermarked images present excellent visual quality, even for small images, because only low-visibility pixels are flipped in the watermark insertion.

(a) Part of original image.

(b) Watermarked image.

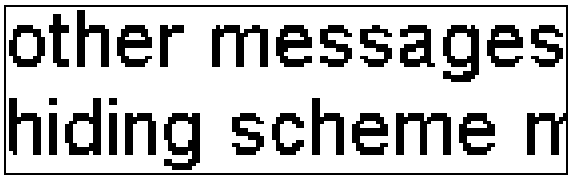
(c) Watermarked image – flipped pixels are printed in color.



(d) Flipped pixels.

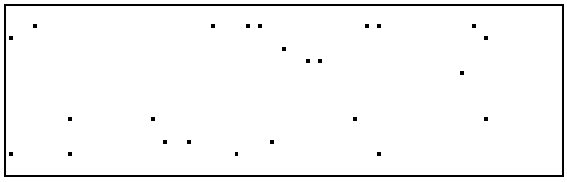
Fig. 6. An image scanned at 300 dpi, with 626×240 pixels, 93 symbols instances and watermarked using AWTCJ with 128-bits long MAC.

(a) Part of original image.



(b) Watermarked image.

(c) Watermarked image – flipped pixels are printed in color.



(d) Flipped pixels.

Fig. 7. An image scanned at 81×81 dpi, with only 194×74 pixels, 56 symbol instances and watermarked using AWTCJ with 128-bits long MAC.

7 Acknowledgements

The authors would like to thank FAPESP and CNPq for the partial financial supports of this work under grants 2003/13752-9 and 305065/2003-3, respectively.

8 References

[1] 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.

[2] 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.

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

[4] H. Y. Kim, "A New Public-Key Authentication Watermarking for Binary Document Images Resistant to Parity Attacks", submmitted to *IEEE Int. Conf. on Image Processing*, (Italy), 2005.

[5] 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.

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

[7] S. V. D. Pamboukian, H. Y. Kim and R. L. de Queiroz, "Watermarking JBIG2 text region for Image Authentication", submmitted to *IEEE Int. Conf. on Image Processing*, (Italy), 2005.

[8] 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.

[9] 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.

[10] I. G. Chun and S. Ha, "A Robust Printed Image Watermarking Based on Iterative Halftoning Method," 2nd Int. Workshop on Digital Watermarking, *Lecture Notes in Computer Science* 2939, pp. 200-211, 2003.

[11] 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.

[12] K. T. Knox and S. Wang, "Digital Watermarks Using Stochastic Screens, Color Imaging: Device-Independent Color, Color Hard Copy, and Graphic Arts II," *SPIE Proc.*, vol. 3018, pp.316-322, Feb. 1997.

[13] N. F. Maxemchuk and S. Low, "Marking Text Documents," *Int. Conf. Image Processing*, vol. 3, pp. 13-17, 1997.

[14] Z. Baharav and D. Shaked, "Watermarking of Dither Halftone Images," *Hewlett-Packard Labs. Tech. Rep.* HPL-98-32, 1998.

[15] 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.

[16] 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.
[17] 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.
[18] D. Boneh, B. Lynn and H. Shacham, "Short signatures from the Weil pairing," Advances in Cryptology - Asiacrypt'2001, *Lecture Notes in Computer Science* 2248, pp. 514-532, 2002.
[19] B. Schneier, *Applied Cryptography*, John Wiley & Sons, 1996.