# Alteration-Locating Authentication Watermarking for Binary Images

Hae Yong Kim and Ricardo Lopes de Queiroz

[1] Escola Politécnica, Universidade de São Paulo, Brazil
hae@lps.usp.br
http://www.lps.usp.br/~{}hae
[2] Universidade de Brasília, Brazil
queiroz@ieee.org

**Abstract.** In image authentication watermarking, hidden data is inserted into an image to detect any accidental or malicious image alteration. In the literature, quite a small number of cryptography-based secure authentication methods are available for binary images. This paper proposes a new authentication watermarking method for binary images. It can detect any visually significant alteration while maintaining good visual quality for virtually all types of binary images (possibly excluding dispersed-dot halftones). As usual, the security of the algorithm lies only on the secrecy of a secret- or private-key. This paper also presents a variation of the proposed scheme that can locate the modified region with good spatial resolution. A possible application of the proposed technique is in Internet fax transmission, i.e. for legal authentication of documents routed outside the phone network.

## 1 Introduction

Classically, steganography (also known as data/information hiding) studies how to hide secret messages in other messages, such that the secret's very existence is concealed. In this paper, data hiding scheme simply means the technique to embed a sequence of bits in a still image and to extract it afterwards without worrying about the confidentiality of the secret's existence.

A watermarking technique makes use of a data-hiding scheme to insert some information in the host image, in order to make an assertion about the image later. Watermarking techniques can be classified as either "robust" or "fragile." Robust watermarks are useful for copyright and ownership assertion purposes. They cannot be easily removed and should resist common image-manipulation procedures such as rotation, scaling, cropping, brightness/contrast adjusting, lossy compression, printing, scanning, etc. On the other hand, fragile watermarks (or authentication watermarks) are easily corrupted by any image processing procedure. However, watermarks for checking the 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.

I.J. Cox et al. (Eds.): IWDW 2004, LNCS 3304, pp. 125–136, 2005.

In the literature, there are many authentication-watermarking techniques (AWTs) for continuous-tone images [27, 26, 22, 23, 17, 2, 13, 24, 3, 5]. Also, there are many techniques for data hiding in binary and halftone images [9, 18, 1, 6, 25, 10, 11, 21, 19]. However, quite a small number of secure AWTs are available for binary and halftone images. We mean by "secure AWT" a scheme where the security does not lie on the secrecy of the algorithm but only on the secrecy of the key. The watermarking algorithm and the fact that an image is watermarked may be made public without compromising the security. Hence, usually a secure AWT relies upon cryptography, and it seems to be very difficult to design a really secure AWT without making use of the solid cryptography theory and techniques. Moreover, a secure AWT must detect *any* visually significant change made to an image. A cryptography-based secure AWT [14, 15] was recently devised for dispersed-dot halftone images but the visual quality for a generic binary image is poor.

In a typical cryptography-based AWT, an authentication signature (AS) is computed from the whole image and inserted into the image itself. 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. An AS contains information about the host image content that may be checked to verify its integrity. 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. For continuous-tone images, many AWTs compute the AS from the image clearing the least significant bits (LSBs) and insert the AS in LSBs. In other words, the host image is divided in two parts: LSBs and the remainder of the image excluding LSBs. Clearly, the LSB-clearing technique cannot be applied to binary images.

In this paper, we propose a new cryptography-based secure AWT for binary images that has good visual quality when applied to generic binary images (excluding dispersed-dot halftones). It can be used in conjunction with secret-key or public/private-key ciphers. We also present a variation of the algorithm that can spatially locate the modifications (besides detecting them). This version does not lose its alteration-locating capability even with image cropping. A possible use of our method is to send faxes and documents over uncontrolled networks and the Internet. In this case, the receiver of a document can verify its integrity for a given originator.

## 2 Data Hiding and Authentication Watermarking

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

**Pixel-wise:** Change the values of (usually pseudo-randomly chosen) individual pixels [10, 11, 7]. 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 [16].

**Component-wise:** Change the characteristics of pixel groups (for example, the position or the area of connected components) [18]. Unfortunately, the success of this approach depends on the type of the host image.

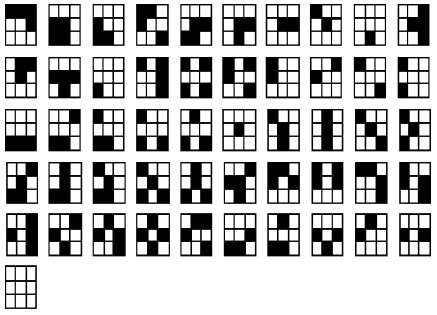
**Block-wise:** Divide the host image into blocks and modify some characteristics of each block [1, 12, 19, 25, 21].

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 recently used to transform a pixel-wise data hiding scheme [10, 11] into an AWT for dispersed-dot halftone images that can detect even single pixel toggling [14, 15]. 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."

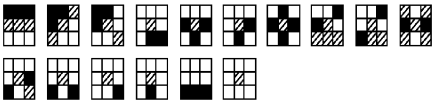
## 3 The Proposed Method

As we noted in previous section, there are some data hiding techniques for binary images. Among them, an interesting technique is template ranking [8, 25], which can be applied to most binary images with excellent visual quality. It can be summarized as follows:

- Divide the image $Z$ to be marked into blocks (e.g. $8 \times 8$).
- The neighborhood of each pixel (usually a $3 \times 3$ template) is analyzed to rate its visual significance. A pixel with low visual significance may change its color with small visual impact. Figure 1 depicts two possible rankings of $3 \times 3$ templates. It seems that the most visually pleasant ranking depends on the nature of the image $Z$ to be marked.
- Insert one bit in each block by forcing the block to have even or odd number of white pixels, to insert bits 0 or 1 respectively. If the block already has the desired parity, it is left untouched. Otherwise, toggle the pixel in the block with the lowest visual significance.



(a) A suggested template ranking [8].



(b) Another rank where hatched pixels match either black or white pixels (the score of a given pattern is that of the matching template with the highest impact).

**Fig. 1.** Two template rankings (in increasing visual significance order from left to right, top to bottom). Mirrors, transposes and reverses of each pattern have the same score. All white or all black patterns have the highest visual impacts

As different blocks may have different quantities of low-visibility pixels, it is suggested to shuffle image $Z$ before embedding data. The template ranking data hiding technique can be employed in a secure AWT for binary images. That was hinted in [25] but not elaborated before. We refer to this technique as AWTR (Authentication Watermarking by Template Ranking). The first version of the AWTR insertion algorithm is:

**Step 1)** Let be given a binary image $Z$ to be marked. Using a pseudo-random number generator with a seed, construct an auxiliary data structure called shuffling vector, so that the image $Z$ can be viewed as a completely shuffled sequence of pixels $\tilde{Z}$. In the secret-key version of our technique, the secret-key is used as the seed of the pseudo-random generator. In the public/private-key version, the seed must be made public.

**Step 2)** Let $n$ be the length of the adopted AS, and $m$ be the number of pixels in each block. Divide the shuffled sequence $\tilde{Z}$ into two regions:
- First region $\tilde{Z}_1$ with $n \times m$ pixels, where the AS is to be stored. This region is subdivided into $n$ blocks with $m$ pixels each. In each block, one bit of the AS will be inserted.
- Second region $\tilde{Z}_2$: the remainder of the shuffled sequence $\tilde{Z}$. The insertion algorithm will compute AS of this region.

**Step 3)** Using a cryptographically secure hashing function $H$, compute the fingerprint of the second region $H = H(\tilde{Z}_2)$. Encrypt the fingerprint $H$ using a secret- or private-key $k$, obtaining an authentication signature $S = E_k(H)$.

**Step 4)** Insert $S$ into the first region, obtaining the watermarked image $Z'$. Insert one bit of $S$ in each block as previously described.

We remark here that AS cannot be made too short without seriously compromising security. A 64-bits-long message authentication code does not with-

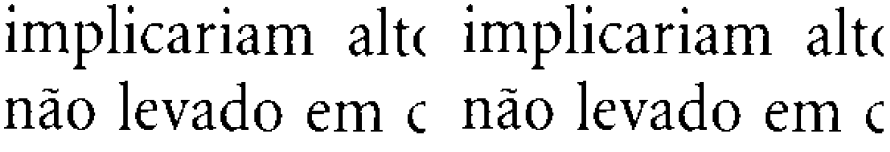
stand a birthday attack [3]. Usually, a message authentication code (MAC) with 128-bits is considered secure. The best known digital signature (DS), RSA, is considered secure with 1024 bits. A newer scheme, DSA, is considered secure with 320 bits. A brand new scheme, BLS, is supposedly secure with only 160 bits [4]. The reader is referred to introductory books on Cryptography for more details (for example, [20]). The verification algorithm of a watermarked image $Z'$ is straightforward:

**Step 1)** Compute the same shuffling vector used in the insertion. Note that in the secret-key version, the secret-key is also the seed of the random number generator and consequently only the owner of the key can reconstruct the shuffling vector. However, in the public-key version, the seed is public and so is the shuffling vector.

**Step 2)** Divide the shuffled sequence $\tilde{Z}'$ into two regions $\tilde{Z}'_1$ and $\tilde{Z}'_2$, in the same way done in the insertion. Compute the fingerprint $H = H(\tilde{Z}'_2)$.

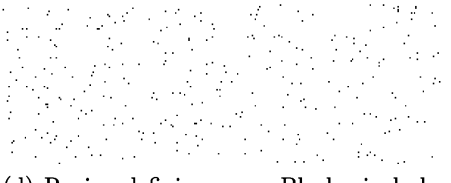
**Step 3)** Extract the authentication signature $S$ stored in $\tilde{Z}'_1$ and decrypt it using the secret- or public-key $k$, obtaining the check data $D = D_k(S)$.

**Step 4)** If $D = H$, the watermark is verified. Otherwise, image $Z'$ has been modified or a wrong key was used.




(a) Part of a page of magazine scanned at 300 dpi which can be considered as a "typical" binary document.

(b) AWTR-marked image (1024 bits embedded, enough for embedding the RSA digital signature).



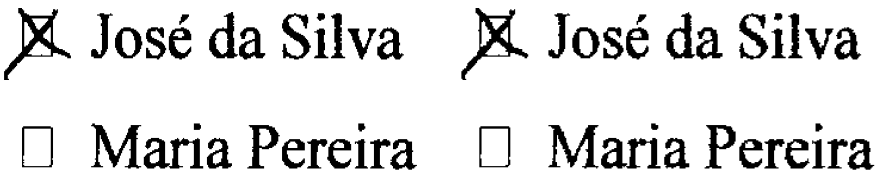
(c) Exclusive-or between figures 2(a) and 2(b).

(d) Region-defining map: Black pixels belong to region 1, while white ones to region 2.

**Fig. 2.** Quality of a "typical" binary document marked with the AWTR

Figure 2(a) depicts a zoom of a magazine page scanned at 300 dpi, which can be considered as a "typical" binary document ($3200 \times 2432$ pixels for the full page). Figure 2(b) is the corresponding image after embedding 1024 bits using 64-pixel blocks. Note in figure 2(c) that only a few pixels have changed their values.

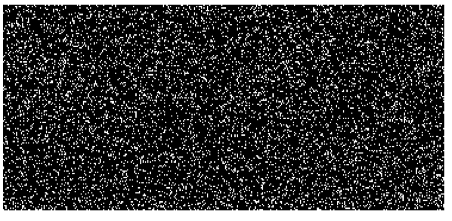
Figure 3(a) shows a very small binary image (160 × 370 pixels) to be watermarked. Figure 3(b) shows the AWTR-watermarked image with 800 bits embedded using 64-pixel blocks. The quality of the marked image is acceptable, even though many bits have been embedded.



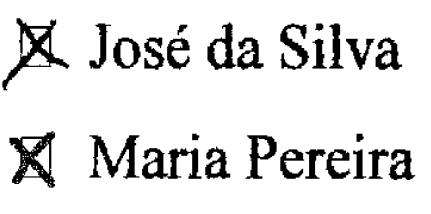

(a) A very small image (160 × 370) to be watermarked.




(b) Image with 800 bits embedded.



(c) Region-defining map: Black pixels belong to region 1, where a parity attack can be mounted.




(d) A fake image generated by a parity attack, undetectable by the first version of the AWTR, but detectable by the second version.

**Fig. 3.** Quality of a very small image marked with the AWTR and the parity attack

## 4 Parity Attack

The proposed method detects any alteration of the second region of a watermarked image, even if one single pixel is toggled. Indeed, the probability of not detecting a modification in this region is only $2^{-n}$ ($n$ is the length of the AS), which can be neglected.

Unfortunately, any alteration that maintains the parities of the blocks in the first region cannot be detected by AWTR. For example, if two pixels that belong to the same block change their values, the parity of this block does not change and this modification will pass undetected. We named this a "parity attack." If the host image $Z$ is large enough, pixels of the first region constitute isolated pixels randomly dispersed in the image and it is unlikely that a malicious attacker will be able to introduce any visually significant alteration by changing only the region-1 pixels (while maintaining the parities of all blocks). For example, in Fig. 2(d), black pixels belong to the region 1 and they are quite dispersed. Thus, no visually meaningful alteration will result by modifying only these pixels. As a rule of thumb, we can consider that no visually significant parity attack can

occur if the number of pixels of the first region is, say, less than 5% of total host image pixels.

However, if image $Z$ is small, region-1 pixels can form contiguous areas. This rises the possibility of a visually meaningful modification that would pass undetected. For example, Fig. 3(c) shows pixels that belong to region 1 in black. Any region-1 pixel can be modified, provided that another pixel in the same block is also modified. To obtain a fake image, a hacker changes a pixel $p$ of block $i$. Then, the hacker looks for the pixel within block $i$ with the lowest visibility and flips its value. Figure 3(d) shows a fake image constructed by repeating this idea. This alteration will pass undetected by AWTR.

Actually, the above described scenario only applies to the public-key version of the AWTR, where the locations of regions 1 and 2, as well as the subdivisions of region 1 into blocks are publicly known. Unfortunately, we could not find any technique to thwart the parity attack for public-key AWTR, except forcing region-1 to be much smaller than the size of the host image. This can be achieved by using small blocks and/or short digital signatures.

For secret-key AWTR, we do not have to worry *much* about a parity attack, because the secret-key is used to generate the shuffling vector. So, an attacker will not know how the watermarked image is divided into regions 1 and 2, and how region 1 is subdivided into blocks. However, we have to pay some attention to this potential attack because the hacker may have many different ways to obtain "clues" about the location of regions and blocks. For example, let us suppose that the hacker has access to a database of original and marked binary documents, all of the same size and all watermarked using the same secret-key. Then the hacker will know that all those pixels whose values are different in the original and the watermarked images belong to region-1 (although it will not be known how region-1 is subdivided into blocks). Nevertheless there are always means to make the division into regions to be image dependent, therefore unique for each image. In any case, in order to minimize the possibility of a parity attack, we suggest the following improvement of step 4 of the secret-key AWTR insertion algorithm:

**Step 4)** Insert S into the first region using the following algorithm, to obtain the watermarked image $Z'$:
For $i \leftarrow 0$ to $n-1$ {
- Insert bit $i$ of $S$ into the $i$-th block forcing it to have odd or even number or white pixels;
- Compute the new AS $S$, feeding the hashing function with the content of block $i$, and encrypting it with the key $k$: $S \leftarrow E_k(H(S, \text{pixels of block } i))$;

}

In this way, block $n-1$ can still suffer a parity attack without being detected. However, if block $n-2$ is modified without modifying its parity, with 50% of chance this modification will be detected. If block 0 is changed (maintaining its parity), there is a probability of $1-2^{-(n-1)}$ of detecting this change. Obviously,

the AWTR verification algorithm must be changed accordingly. Unfortunately, this idea cannot be used with the public-key AWTR, because the digital signature must be completely extracted before decrypting it with the public-key.

## 5 Locating Alterations

The AWTR can be transformed into an authentication watermarking capable of spatially locating the altered regions. Let us call this technique AWTRAL (Authentication Watermarking by Template Ranking with Alteration Locating). The naive idea of dividing the host image into sub-images and authenticating independently each sub-image is not safe. Wong [22, 23] has proposed a similar idea to authenticate continuous-tone images and it was demonstrated later to be subject to many kinds of attacks, including the "cut and paste," "vector quantization counterfeiting" and "birthday" attacks [2,13,24,3]. A few works [24, 3,5] present different approaches to protect the host image against such attacks. Wong and Memon [24] propose a technique where the AS of each sub-image is dependent of its index and a unique identifier of the host image. However, the image identifier has to be somehow stored outside of the host image, what is a significant inconvenience. Barreto et al. [3] present a technique called "hash block chaining" that uses the contexts of sub-images and a non-deterministic digital signature to obtain the security. Using this technique, there is no more need to use an image identifier. However, the obtained spatial resolution is twice the size of each sub-image. Celik et al. [5] propose a hierarchical watermarking. The spatial resolution of this scheme is nearly the size of each sub-image. It does not lose its alterations-locating capability even with an image cropping: it can use a "sliding window" to resynchronize with sub-images. We will use the hierarchical watermarking with two layers to construct the AWTRAL. Two layers are sufficient to obtain the desired security.

The idea of AWTRAL is to divide the image to be marked into sub-images (say, with $128 \times 128$ pixels). Then, we watermark independently each sub-image with secret-key AWTR (first layer), and watermark the whole image (composed by all AWTR-marked sub-images) with another public- or secret-key AWTR (second layer). Note that if the host image is not large enough to hinder parity attacks, secret-key AWTR must be used in both watermark layers. Using secret/public-key AWTRAL, anyone can verify if the marked image is authentic using the public-key. If an image is found to be fraudulent, the owner of the secret-key can spatially locate the alterations to help discovering the intentions of the hacker. We describe below the secret/public-key AWTRAL:

**Step 1)** Let be given a binary image $Z$ to be marked. Pseudo-randomly shuffle $Z$, obtaining the shuffled sequence of pixels $\tilde{Z}$.

**Step 2)** Divide the sequence $\tilde{Z}$ into two regions:

- Region $\tilde{Z}_A$ with $n_2 \times m$ pixels, where the second-layer DS is to be stored ($n_2$ is the length of adopted DS and $m$ is the size of each block).
- Region $\tilde{Z}_{BC}$: the remainder of $\tilde{Z}$ from where the DS is to be computed. This region will be further subdivided into regions $\tilde{Z}_B$ and $\tilde{Z}_C$.

**Step 3)** Divide the original non-shuffled image $Z$ into sub-images (say, with $128 \times 128$ pixels) $Z_1$, $Z_2$, ..., $Z_l$.

**Step 4)** Pseudo-randomly shuffle each sub-image $Z_i$, obtaining $\tilde{Z}_i$, and divide it into three sub-regions:
- Sub-region $\tilde{Z}_{iA}$, constituted by the pixels of sub-image $\tilde{Z}_i$ that belong to the region $\tilde{Z}_A$ determined in step 2.
- Sub-region $\tilde{Z}_{iB}$ with $n_1 \times m$ pixels, where the first-layer MAC is to be stored ($n_1$ is the length of adopted MAC and $m$ is the size of each block).
- Sub-region $\tilde{Z}_{iC}$: the remainder of $\tilde{Z}_i$ from where the MAC is to be computed.

**Step 5)** For each shuffled sub-image $\tilde{Z}_i$, compute the fingerprint of sub-region $\tilde{Z}_{iC}$: $H_i = H(\tilde{Z}_{iC})$. Encrypt the fingerprint $H_i$ with a secret-key, obtaining the MAC $S_i = E_k(H_i)$. Insert $S_i$ into sub-region $\tilde{Z}_{iB}$, as described in sections 3 and 4.

**Step 6)** Compute the fingerprint of region $\tilde{Z}_{BC}$: $H = H(\tilde{Z}_{BC})$. Encrypt the fingerprint $H$ with the private-key, obtaining the digital signature $S = E_k(H)$. Insert $S$ into region $\tilde{Z}_A$, as described in sections 3 and 4.

The AWTRAL verification algorithm is:

**Step 1)** Given an AWTRAL-marked image $Z'$, shuffle it as in the insertion, obtaining the shuffled sequence $\tilde{Z}'$.

**Step 2)** Divide the sequence $\tilde{Z}'$ into regions $\tilde{Z}'_A$ and $\tilde{Z}'_{BC}$, as before. Compute the fingerprint $H$ of $\tilde{Z}'_{BC}$.

**Step 3)** Extract the DS $S$ stored in $\tilde{Z}'_A$ and decrypt it using the public-key $k$, obtaining the check data $D = D_k(S)$.

**Step 4)** If $D = H$, the watermark is verified and no further processing is necessary. Otherwise, image $Z'$ was modified and the following steps can determine the altered locations.

**Step 5)** Divide the watermarked non-shuffled image $Z'$ into sub-images $Z'_1$, $Z'_2$, ..., $Z'_l$, as in the insertion.

**Step 6)** Shuffle each sub-image $Z'_i$, obtaining $\tilde{Z}'_i$, and divide it into three sub-regions $\tilde{Z}'_{iA}$, $\tilde{Z}'_{iB}$ and $\tilde{Z}'_{iC}$, as before.

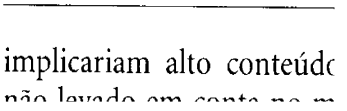
**Step 7)** For each shuffled sub-image $\tilde{Z}'_i$, compute the fingerprint $H_i$ of sub-region $\tilde{Z}'_{iC}$. Extract the MAC $S_i$ stored in sub-region $\tilde{Z}'_{iB}$ and decrypt it, obtaining the check data $D_i = D_k(S_i)$. If $H_i = D_i$, the watermark of sub-image $Z'_i$ is verified.

Let us make three remarks on the AWTRAL algorithm. First, the above-described algorithm loses its capability of locating alterations after row or column cropping. Celik et al. [5] suggest using a sliding window to obtain a cropping-surviving watermark. Step 5 of the verification algorithm may be modified as follows to locate alterations even after image cropping:

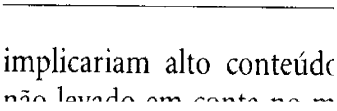
**Step 5)** Slide a window (of the same size as the sub-images used in the AWTRAL insertion) over the non-shuffled image $Z'$ to obtain all possible sub-images $Z'_1$, $Z'_2$, ..., $Z'_l$ within $Z'$.

Second, there may be some sub-images almost completely white (black) with only a few black (white) pixels. These sub-images have few low-visibility pixels and the watermark insertion may make visible salt-and-pepper noise. We suggest not watermarking neither verifying these sub-images, because it is not possible to introduce any visually significant alteration without significantly increasing the number of black (white) pixels. And if the number of black (white) pixels increases significantly, the MAC verification will detect the forgery.

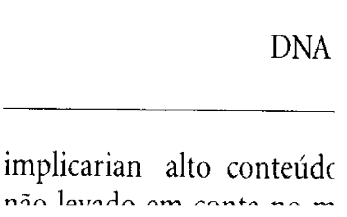
Third, the first-layer MAC can be shorter than the usual 128 bits, because the AWTRAL cannot be assaulted by a conventional birthday attack. If a birthday attack takes place and a sub-image is replaced by another fake sub-image, the second-layer DS will detect this alteration. However, the AWTRAL can be assaulted by an "improved birthday attack" similar to that described in [3]. It consists in replacing simultaneously two sub-images by two fake sub-images such that the two sub-images' MACs and the DS of the whole image remain valid. Hence, the first-layer MAC must be longer than 64 bits: we recommend using a 96-bit or longer MAC.



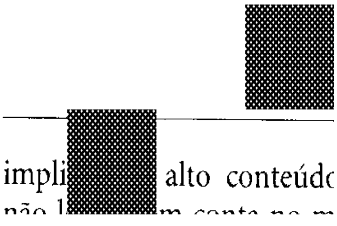

(a) Part of a page of a magazine scanned at 300 dpi.




(b) AWTRAL-watermarked image.




(c) A faked image: A letter "m" was transformed into a letter "n" and the word "DNA" was inserted.




(d) Locating the alterations.

**Fig. 4.** The use of the alteration-locating watermark AWTRAL

Figure 4 illustrates the AWTRAL. Figure 4(a) is part of a page of a magazine which was scanned at 300 dpi yielding $3200 \times 2432$ pixels per page. This image was marked by the AWTRAL, resulting in Fig. 4(b). The host image was subdivided into 475 sub-images each with $128 \times 128$ pixels. Each sub-image was marked with a 96-bits-long MAC, using blocks of 64 pixels. A 1024-bits-long DS, also using blocks of 64 pixels, was used to protect the whole image. Almost completely white (black) sub-images, with 6 or less black (white) pixels, were not marked. Figure 4(c) is a fake image, where a letter "m" was switched by the letter "n"

and the word "DNA" was inserted. Figure 4(d) shows the result of the AWTRAL verification algorithm, where the modified sub-images are clearly marked.

## 6 Conclusions

This paper has proposed a new cryptographically secure authentication watermarking technique for binary images (AWTR). The proposed technique is suitable to watermark most binary images with excellent visual quality. We also described a variant of the AWTR that can locate the alterations, besides detecting them. The AWTR can be used in trusted FAX machines, i.e., to electronically sign binary documents. Further research is necessary to adapt the method to scanned documents.

## Acknowledgements

The authors would like to express their gratitude to Dr. Paulo S. L. M. Barreto for warning us about the possibility of an improved birthday attack against the AWTRAL. One of the authors, H. Y. Kim, 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.

## References

1. Z. Baharav and D. Shaked, "Watermarking of Dither Halftone Images," Hewlett-Packard Labs. Tech. Rep. HPL-98-32, 1998.
2. P. S. L. M. Barreto and H. Y. Kim, "Pitfalls in Public Key Watermarking," Sibgrapi – Brazilian Symp. Computer Graphics and Image Processing, pp. 241-242, 1999.
3. P. S. L. M. Barreto, H. Y. Kim and V. Rijmen, "Toward a Secure Public-Key Blockwise Fragile Authentication Watermarking," IEE Proc. Vision, Image and Signal Processing, vol. 149, no. 2, pp. 57-62, 2002.
4. 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.
5. 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.
6. Y.-Y. Chen, H.-K. Pan and Y.-C. Tseng, "A Secure Data Hiding Scheme for Binary Images," IEEE Symposium on Computers and Communications, pp. 750-755, 2000.
7. I. G. Chun and S. Ha, "A Robust Printed Image Watermarking Based on Iterative Halftoning Method," 2nd Int. Workshop on Digital Watermarking, Lecture Notes on Computer Science 2939, pp. 200-211, 2003.
8. R. de Queiroz and P. Fleckenstein, "Object Modification for Data Embedding through Template Ranking," Xerox Invention Proposal, 1999.
9. M. P. Deseilligny and H. Le Men, "An Algorithm for Digital Watermarking of Binary Images, Application to Map and Text Images," available at www-ima.enst.fr/~maitre/tatouage/MPdS_HK.ps, 1998.

10. M. S. Fu and O. C. Au, "Data Hiding by Smart Pair Toggling for Halftone Images," IEEE Int. Conf. Acoustics, Speech and Signal Processing, vol. 4, pp. 2318-2321, 2000.
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. H. Z. Hel-Or, "Watermarking and Copyright Labeling of Printed Images," Journal of Electronic Imaging, vol. 10, no. 3, pp. 794-803, 2001.
13. M. Holliman and N. Memon, "Counterfeiting Attacks on Oblivious Block-wise Independent Invisible Watermarking Schemes," IEEE Trans. Image Processing, vol. 9. no. 3, pp. 432-441, 2000.
14. H. Y. Kim and A. Afif, "Secure Authentication Watermarking for Binary Images," in Proc. Sibgrapi – Brazilian Symp. on Comp. Graph. and Image Proc., pp. 199-206, 2003.
15. H. Y. Kim and A. Afif, "Secure Authentication Watermarking for Halftone and Binary Images," to appear in Int. J. Imaging Systems and Technology.
16. 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.
17. C. T. Li, D. C. Lou and T. H. Chen, "Image Authentication and Integrity Verification via Content-Based Watermarks and a Public Key Cryptosystem," IEEE Int. Conf. Image Processing, 2000, vol. 3, pp. 694-697.
18. N. F. Maxemchuk and S. Low, "Marking Text Documents," Int. Conf. Image Processing, vol. 3, pp. 13-17, 1997.
19. 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.
20. B. Schneier, Applied Cryptography, John Wiley & Sons, 1996.
21. 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.
22. P. W. Wong, "A Watermark for Image Integrity and Ownership Verification," IS&T PIC Conference, (Portland, OR), May 1998 (also available as Hewlett-Packard Labs. Tech. Rep. HPL-97-72, May 1997).
23. 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).
24. P. W. Wong and N. Memon, "Secret and Public Key Image Watermarking Schemes for Image Authentication and Ownership Verification," IEEE Trans. Image Processing, vol. 10, no. 10, pp. 1593-1601, 2001.
25. 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.
26. M. M. Yeung and F. Mintzer, "An Invisible Watermarking Technique for Image Verification," IEEE Int. Conf. Image Processing, 1997, vol. 1, pp. 680-683.
27. J. Zhao and E. Koch, "Embedding Robust Labels into Images for Copyright Protection," Proc. Int. Cong. Intellectual Property Rights, Knowledge and New Technologies, 1995, pp. 242-251.