Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 681–690
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

An efficient image authentication method based on Hamming code

Chi-Shiang Chan[a], Chin-Chen Chang[b,*]

[a]Department of Computer Science and Information Engineering, National Chung Cheng University, Chaiyi 621, Taiwan, ROC
[b]Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan, ROC

Received 4 December 2005; received in revised form 9 December 2005; accepted 15 May 2006

Abstract

Image authentication has come through a history of several years. However, up to the present time, most mainstream image authentication schemes are still unable to detect burst bit errors. Moreover, the capability of recovering tampered pixels in detail (complex) areas has not been very satisfactory either. In this paper, we offer to combine the Hamming code technique, Torus automorphism and bit rotation technique to do tamper proofing. According to our experimental results, our new hybrid method can effectively eliminate burst bit errors, and our recovered pixels in detail areas can actually gain very high clarity. The results show that our scheme is quite a practical method, which is quite able to detect and recover tampered areas.

Keywords: Image authentication; Tamper proofing; Hamming code; Torus automorphism

1. Introduction

As a result of the rapid advancement of computer technology, digitized media of all kinds, especially digital images, have become more and more popular. Because of the easy-to-tamper nature of digitized media, the urgent need for media security and copyright protection mechanisms such as image watermarking have been growing. In reality, image watermarking techniques have come through a history of several years, and they have gone down different routes. For example, in terms of robustness, image watermarking techniques can be classified as either robust image watermarking techniques [1–6] or fragile ones [7–12].

The purpose of robust image watermarking is to protect the ownership of the digital image. In a typical robust image watermarking scheme, a watermark is embedded into the digitized image. Once there is any dispute over the ownership of the image, whether or not the digitized image has been tampered, the robust image watermarking system should be able to extract the watermark from modified image. The ownership can be verified by revealing the extracted watermark. Since robust image watermarking is used to protect the ownership, it has to offer the power of extracting the watermark after any kind of tampering.

On the other hand, fragile image watermarking mainly deals with image authentication. The purpose of image authentication is used to ensure the integrity of the contents of the digital image. To do that, a good fragile image watermarking scheme should be capable of sensing tampering of any kind with the digital image.

Image authentication can be further divided into two splits: the digital signature approach [8–11] and the watermark-based approach [7,12]. Image authentication schemes that follow the digital signature approach extract a set of features from the digital image and keep the authentication message as independent data. When it is time to check whether an authenticated image has been tampered, the authentication message will be handed out for verification. Because the authentication message is independent data, any piece of authentication message will not be modified when the digital image is tampered. Therefore, this kind of image authentication is expected not only to resist malicious modifications but also to tolerate modifications caused by such image-processing incidents as JPEG compression or blurring. The drawback here is that the owner must keep the authentication message very carefully for future use.

* Corresponding author. Tel.: +886 4 24517250; fax: +886 5 2720859.
E-mail addresses: cch@cs.ccu.edu.tw (C.-S. Chan), ccc@cs.ccu.edu.tw (C.-C. Chang).

0031-3203/$30.00
doi:10.1016/j.patcog.2006.05.018

Image authentication schemes that follow the watermark-based approach, on the other hand, embed the authentication message into the digital image content. For content authentication, the authentication message must be extracted and verified. Therefore, this kind of image authentication scheme is supposed to be able to extract the authentication message after the content of the digital image is changed. According to the extracted authentication message, it can detect any malicious modification. Watermark-based image authentication schemes usually embed the authentication message into the raw data because the capacity there is big enough.

Currently, the existing fragile image watermarking schemes are capable of detecting any malicious modification. However, most of them have two drawbacks. The first one is that those techniques cannot detect the so-called burst bit errors that happen when the digital image is being transmitted on the Internet. Burst bit errors may seriously harm the watermarked image when they cause bit value changes in the most significant bits (MSB). The second drawback is that although those techniques can detect a tampered area, they do not seem perfectly capable of recovering the tampered area. Generally speaking, fragile image watermarking schemes can successfully recover tampered bits in non-detail (smooth) areas. However, if the tampered area is a detail (complex) area, most schemes do not seem to be able to do a good recovery job.

In this paper, we shall propose a new method to overcome those two drawbacks. Our new method consists of three components: the Hamming code, Torus automorphism and bit rotation. The Hamming code is used to produce parity check data as the authentication message. Torus automorphism is used to spread the authentication message around. As for the final component, bit rotation, it is used to rotate the authentication message so as to improve the security. The Hamming code can help detect and recover burst bit errors. Moreover, we can recover tampered detail areas according to the authentication message. Our experimental results will demonstrate later that our new scheme can actually live up to our high standards and offer quality protection to digital images.

The rest of this paper is organized as follows. To begin with, we shall review the Hamming code in Section 2. Then, we will continue to present our method in Section 3. In Section 4, we shall offer our experimental results to demonstrate the effectiveness of our new method. Finally, the conclusions will be given in Section 5.

2. Related works

First of all, let us briefly review the Hamming code and see how it functions. The Hamming code [13] is a kind of forward error correction (FEC). That means the receiver has ability to correct a transmission error according to the code itself. Therefore, it comes in quite handy when there are data to be transmitted in a noisy environment. To have this ability, some bits must be appended to the original data. The appended bits are called parity check bits. Generally speaking, the Hamming code has the ability to correct single-bit errors and detect two-bit errors. The basic idea of the Hamming code is the concept of parity check.

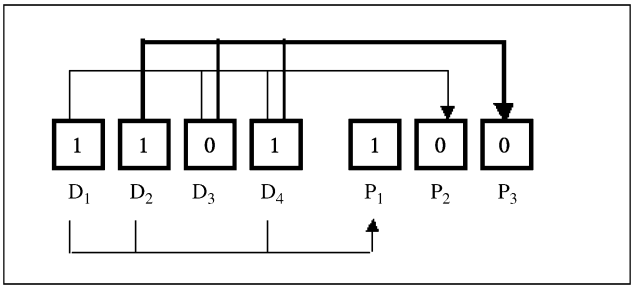

Fig. 1. Data bits and parity check bits.

The number of parity check bits there should be is determined by the Hamming inequality rule, which will not be discussed further. According to the Hamming inequality rule, however, we know that the number of parity check bits for four-bit data is three, which is commonly referred to as the (7,4) Hamming code. With the parity check bits appended to the data bits, the whole thing is called the Hamming code word, and the total number of bits in a (7,4) Hamming code word is seven.

We now introduce the values of the parity check bits for the (7,4) Hamming code and how they can be used to correct errors. Let us refer to the example in Fig. 1. There is some data with four data bits (D_1, D_2, D_3, D_4) and three check parity bits (P_1, P_2, P_3). The lines indicate the relationships between the data bits and the three parity check bits. For example, the value of parity check bit P_1 is related to data bits D_1, D_2 and D_4. Similarly, parity check bit P_2 is related to data bits D_1, D_3 and D_4. Now, how do we decide the values of the parity check bits? In the first place, the final goal is to let the parity check bit and its related data bits achieve "Even Parity". The meaning of "Even Parity" is that the number of "1" bits is even.

Now, let us discuss the reason why the Hamming code can correct one-bit errors. Each overlapping circle in Fig. 2 represents a parity check bit and its related data bits. In this case, the bits in each circle must have "Even Parity". It is obvious that any change of bits will break the property of "Even Parity" for some circles. For example, changing the value of D_4 will expel all the three circles out of the "Even Parity" family. When this situation occurs, it indicates that there is an error bit in the value of D_4. On the other hand, changing the value of D_3 breaks the property of "Even Parity" on two circles. If only one circle disobeys the "Even Parity" rule, then we can determine that the error occurs on the parity check bit.

Following the same example in Fig. 1, we assume that an error occurs in D_4; that is, the value of D_4 changes from 1 to 0. This results in the three cycles shown in Fig. 3.

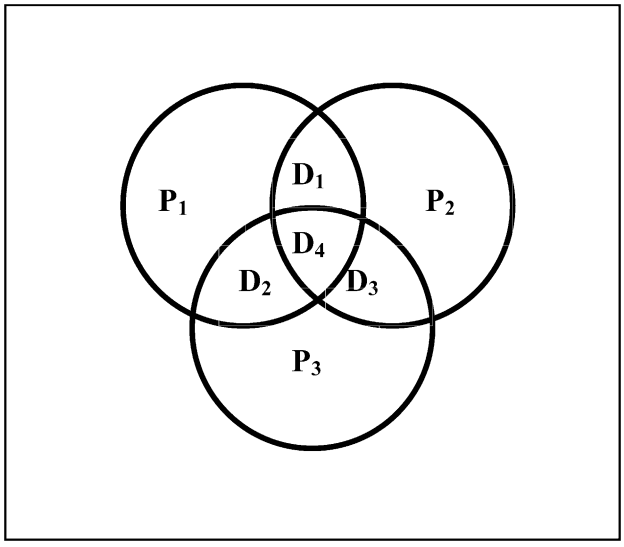

Fig. 2. The relationship between data bits and parity check bits.

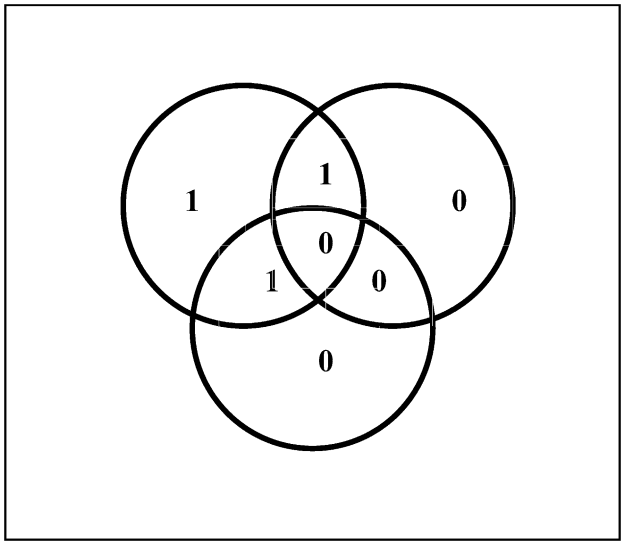

Fig. 3. An error occurs in D_4.

Please note that the number of "1" bits in each cycle is odd. If this situation is caused by one-bit error, then we know that all the cycles will obey the "Even Parity" rule after the value change of D_4 from 0 to 1. This way, we can correct one-bit error from the Hamming code word itself.

3. The proposed method

In this section, we shall present our new method that takes in the Hamming code to do tamper proofing. There are three procedures we have in our method. They are the embedding procedure, the detecting procedure and the recovering procedure. In the embedding procedure, the necessary authentication data gets embedded into the host image. In the detecting procedure, the authenticated image is put through the tamper proofing process. The final procedure, namely the recovering procedure, is an attempt to recover the pixels in the areas that have been tampered. The details of our proposed scheme are as follows.

3.1. The embedding procedure

In this subsection, we shall introduce our embedding procedure. Three components we have put together to build up the embedding procedure: the Hamming code, Torus automorphism and bit rotation. First, we produce parity check bits for each pixel by using the Hamming code rule. Then, for the parity check bits, we use Torus automorphism to decide which pixels should be the carriers of the parity check bits. The final component, bit rotation, is used to rotate the check bits before they are embedded so that the security can be further improved. The following is how the three components work.

First of all, here in this place, let us discuss how the Hamming code can be used in our method. To begin with, let us construct a (7,4) Hamming code for each pixel. In a pixel, there are MSB and least significant bits (LSB). The pixel value changes dramatically when the MSB are changed. However, the pixel value only changes a little when we change the LSB. The main concept of our method is that we produce three check bits for the four MSB and embed the three check bits back to the least three significant bits.

For example, suppose we have a pixel whose value is 208. Its binary representation is $(11010000)_2$. The four MSB are $(1101)_2$. According to the (7,4) Hamming code production rule, the three parity check bits are $(100)_2$. Then, replacing the three LSB with the three parity check bits, we change the original pixel to $(11010100)_2$. If an error occurs in the set of four MSB of the pixel, the pixel value would change greatly. In this case, we can use the three LSB to correct the error in the pixel value.

It seems reasonable to embed the parity check bits into the very pixel they come from. However, there exists a problem here. The purpose of our proposed method is not only error bit correction and tamper proofing but also tampered area recovery. However, if the parity check bits for a pixel are embedded into the pixel itself, both the data bits and parity check bits will probably get destroyed in a tampered area. In order to solve this problem, the parity check bits of a pixel, called pixel A, must be embedded into another pixel, called pixel B. Now it is time for Torus automorphism because we need to decide exactly where this pixel B is so that the parity check bits can be embedded and later extracted out.

Torus automorphism [14] was proposed by Voyatzis and Pitas in 1996. Generally speaking, Torus automorphism is a pretty good tool to do permutation in two dimensions. The Torus automorphism formula is below:

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ k & k+1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \bmod N. \tag{1}$$

The variables x_i and y_i represent the original row and column for the ith pixel, respectively; on the other hand,

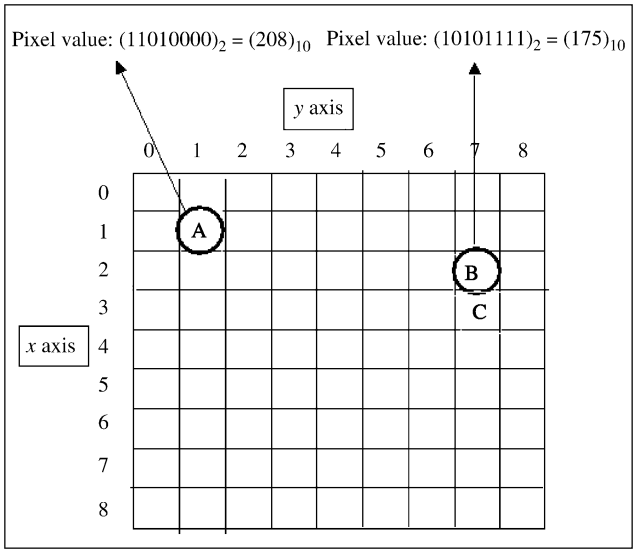

Fig. 4. An example of how Torus automorphism works.

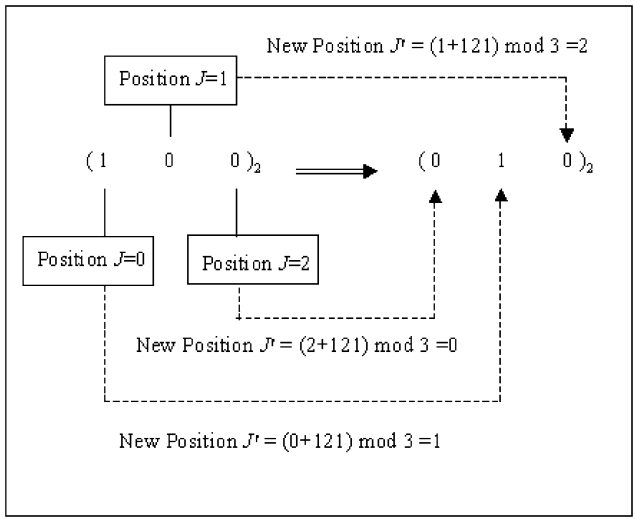

Fig. 5. An example of the bit rotation procedure.

x_i' and y_i' represent the new row and column for the ith pixel, respectively. The height and width of the image are both N. The value k can be taken as a key. According to the key, for each pixel, we can calculate its new location from its original location. When Torus automorphism is used, the parity check bits of a pixel can be embedded into the other pixel in the new position.

For example, assume that the value of N is 9 and the value of k is 3. In Fig. 4, the value of pixel A in the position (1, 1) is 208. The four MSB are $(1101)_2$, and the parity check bits are $(100)_2$. According to the rule of Torus automorphism, the new coordinates for coordinates (1,1) are (2,7), which indicates the position where pixel B is supposed to be. Then, the next thing we do is embedding the check bits $(100)_2$ into pixel B. The original value of pixel B is 175, and its binary representation is $(10101111)_2$. We directly replace the three LSB with the check bits of pixel A. After that, the binary representation of pixel B becomes $(10101100)_2$. Of course, pixel A will also become a carrier for another pixel, that is, pixel C, in our example. The parity check bits of pixel C will be embedded into pixel A.

Torus automorphism does not go without its drawback, though. Let us check out the formula of Torus automorphism first. In fact, the value of k is bounded by the value of N. That is, if we set the key k as x, then the new position indicated by Torus automorphism is the same as when the key k is set as $x + N$. This means the value of the key k falls in the range from 1 to $N - 1$. For each pixel, we can calculate its parity check bits from the four MSB. Since we know what the parity check bits of each pixel look like, we can try the k value from 1 to $N - 1$ to check whether the values of parity check bits are the same as the LSB values of the new position values. Once they match, we can get the right k value.

Because everyone can get the key k, any counterfeiter can calculate the new parity check bits for all modified pixels and embed them into the correct positions according to the Torus automorphism rule and the key k. In order to solve this problem, the parity check bits must go through the procedure of bit rotation before they are embedded into the new positions. The function of bit rotation is to rotate the order of the parity check bits. Here, we must use the second key, called k_2. The second key k_2 is taken as a seed to produce a sequence of random numbers, $R_1, R_2, \ldots, R_{N\times N}$. Before the parity check bits are embedded into the pixel, the bit position of the parity check bits must rotate according to the value of the random number. The rotation formula is as follows:

$$J' = (J + R_i) \bmod M. \tag{2}$$

R_i is the random number assigned to the ith pixel. J represents the order of the parity check bits, and J' denotes the new order of the parity check bits. Finally, M is the number of bits to rotate.

Let us take the same example in Fig. 4. The number of parity check bits is 3, so the value of M is 3. The parity check bits to embed into pixel B are $(100)_2$. We assume that the random number R_i here is 121. After the functioning of the bit rotation procedure, the result is like what we see in Fig. 5. The rotated bits are $(010)_2$. The rotated bits will be embedded in pixel B instead of $(100)_2$ in Fig. 4. Therefore, the final binary representation for pixel B becomes $(10101010)_2$.

3.2. The detecting procedure

Image modifications can happen either because of the noisy environment or malicious tampering.

A sequence of data bits may be affected by noise when transmitted on the Internet. Under such circumstances, if only a one-bit error occurs among these four MSB and three LSB, then it is guaranteed that the error bit can be found out and corrected, and the job can be done simply by the Hamming code itself alone.

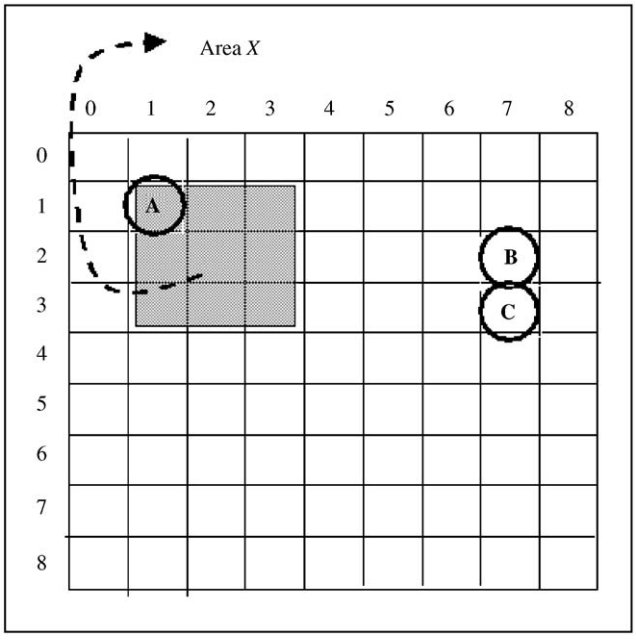

Fig. 6. An example for tampering proof.

Fig. 8. The test image.

Data Bits	Parity Check Bits	The Decimal Value
0 0 0 0	0 0 0	$(0)_{10}$
0 0 0 1	1 1 1	$(7)_{10}$
0 0 1 0	0 1 1	$(3)_{10}$
0 0 1 1	1 0 0	$(4)_{10}$
0 1 0 0	1 0 1	$(5)_{10}$
0 1 0 1	0 1 0	$(2)_{10}$
0 1 1 0	1 1 0	$(6)_{10}$
0 1 1 1	0 0 1	$(1)_{10}$
1 0 0 0	1 1 0	$(6)_{10}$
1 0 0 1	0 0 1	$(1)_{10}$
1 0 1 0	1 0 1	$(5)_{10}$
1 0 1 1	0 1 0	$(2)_{10}$
1 1 0 0	0 1 1	$(3)_{10}$
1 1 0 1	1 0 0	$(4)_{10}$
1 1 1 0	0 0 0	$(0)_{10}$
1 1 1 1	1 1 1	$(7)_{10}$

Fig. 7. The Hamming codeword.

Fig. 9. The authenticated image.

As for the solution to second kind of modification problem, namely malicious tampering, in our method, we draw out the area that has been modified in the first place. Let us see an example in Fig. 6. We assume that the gray area, called area *X*, is a modified area where pixel *A* belongs. Note that the parity check bits of pixel *A* are embedded into pixel *B*, and the parity check bits of pixel *C* are embedded into pixel *A*.

Since pixel *A* has been modified, the recalculated parity check bits of modified pixel *A* are not the same as those embedded in pixel *B*. Therefore, both pixels *A* and *B* will be taken as modified pixels. The situation is the same for pixel *C* because its parity check bits are embedded in pixel *A*. In other words, pixel *C* will also be taken as a modified pixel. Throughout the whole image, we check the pixels one by one and mark all the modified pixels. Observing the final result, we can see that all the pixels in area *X* are marked as modified pixels. At the same time, there are also other areas with isolated pixels marked as modified pixels.

Fig. 10. The resultant image. (a) The image affected by noise. (b) The image corrected by our method.

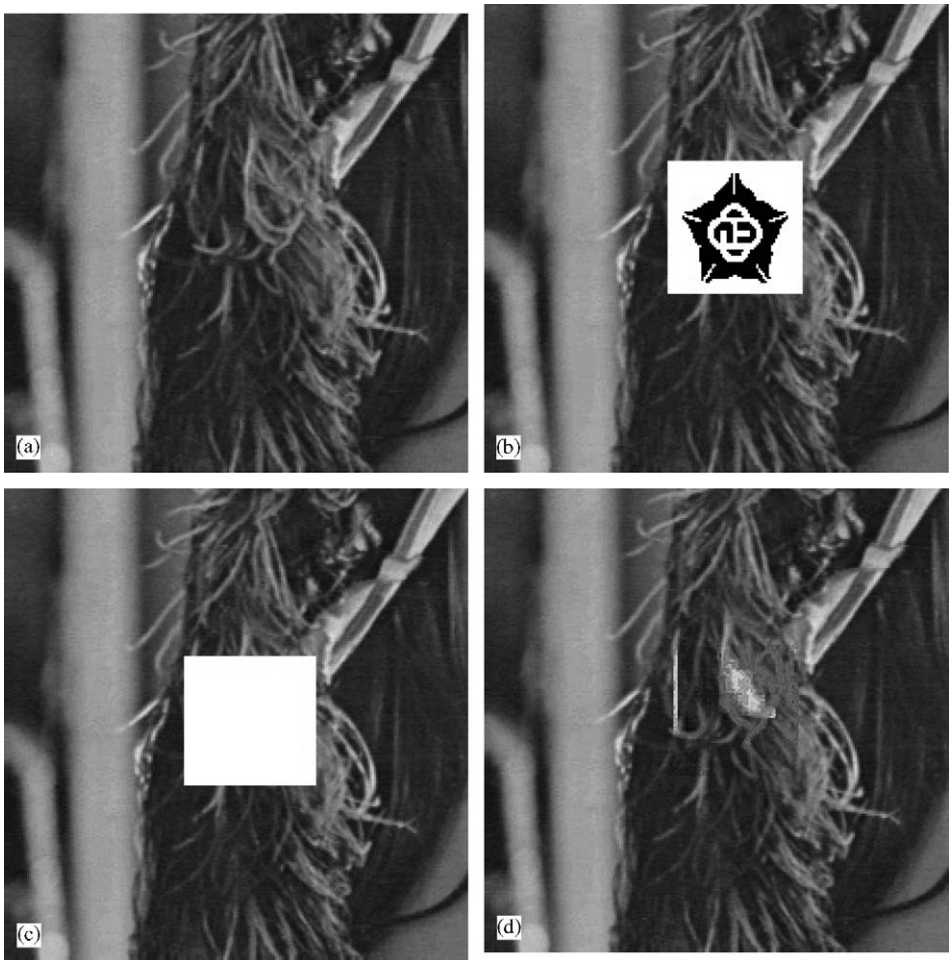

Fig. 11. The resultant partial image for the detail area. (a) The partial image of the authenticated image. (b) The partial image with tampered area. (c) The partial image with detected area. (d) The partial image with recovered area.

Because obviously there exist some faulty judgments such as pixel B and pixel C, we must try to eliminate them. In our method, we use morphological operations [15] to achieve our goal. Morphological operations include two kinds of operations: erosion operations and dilation operations. With the help of these erosion operations and dilation operations, the isolated points can be eliminated. Finally, only the pixels in area X are marked as modified pixels.

3.3. The recovering procedure

After the detecting procedure, we have got some pixels marked as modified pixels. The next thing to do here is to try to recover the value of the modified pixels. Since the pixel values in area X have been modified, there is no information left in the pixels themselves that can help us recover them. However, the parity check bits for the pixels in area X are

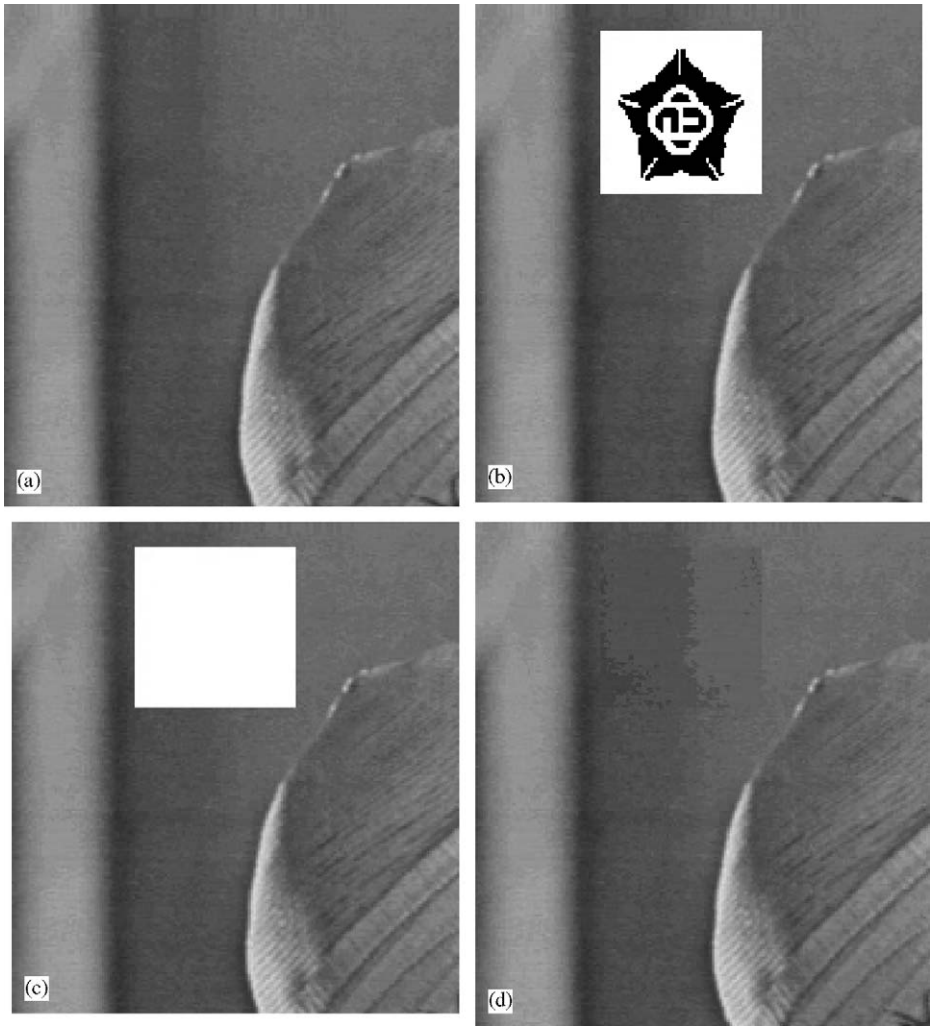

Fig. 12. The resultant partial image for the non-detail area. (a) The partial image of the authenticated image. (b) The partial image with tampered area. (c) The partial image with detected area. (d) The partial image with recovered area.

not necessarily modified. For the rest of this subsection, we shall discuss how our new method will recover a modified area either when the parity check bits still survive or when they have also been modified.

First, suppose we are in the situation where both the data bits and their corresponding parity check bits have been destroyed. That is, both pixel A and pixel B in Fig. 6 are in modified area X. In this case, we use the pixel predictive scheme of JPEG-LS to do prediction. The predictive formula goes as follows:

$$d = \begin{cases} \min(a, b) & \text{if } c \geqslant \max(a, b), \\ \max(a, b) & \text{if } c \leqslant \min(a, b), \\ a + b - c & \text{otherwise.} \end{cases} \quad (3)$$

The parameter d represents the final predicted value for the target pixel. As for a and b, they stand for the values of the pixels on the left and on top of the predicted pixel, respectively. Finally, c denotes the pixel on the top-left of the predicted pixel.

In the other case where the parity check bits still survive, we can use them to guess the pixel values of the modified pixels. Taking the data bits and parity check bits in Fig. 7 for example. We can observe some relations between the two sets of values that help guess the pixel values.

We first divide the (7,4) Hamming code into two groups according to the value of the MSB of the data bits. In each group, the decimal values of the parity check bits are unique. That is, in a group, there are no two different data bits that have the same decimal value. Based on this property, if the value of the MSB of the data bits can be decided, the pixel value can be pointed out according to the value of the parity check bits.

How can we decide the value of the MSB of the data bits, then? The MSB of the data bits represents the MSB of the original pixel. If the MSB is 1 that means the value of the original pixel is larger than 128. Otherwise, the value of the original pixel is smaller than 128. In our recovering procedure, we use Formula (3) to decide whether the value is

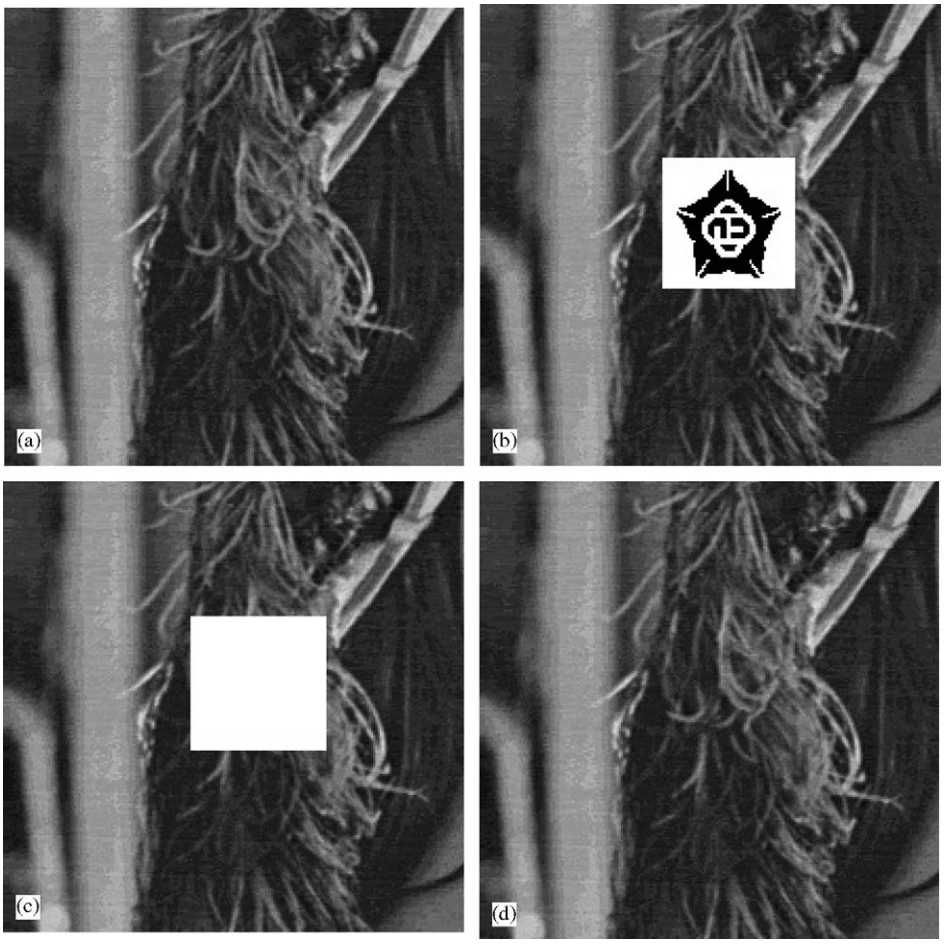

Fig. 13. The resultant partial image for the detail area. (a) The partial image of the authenticated image. (b) The partial image with tampered area. (c) The partial image with detected area. (d) The partial image with recovered area.

larger or smaller than 128. This way, since we have decided the value of MSB, we can recover the pixel value according to the value of the parity check bits.

Let us consider the example in Fig. 6. Following Formula (3), we can predict whether the value of pixel A is larger or smaller than 128. After deciding the MSB of pixel A, we extract the parity check bits of pixel A from pixel B, and the four MSB of pixel A can be decided. Note that the three LSB of pixel A are the parity check bits of pixel C. Therefore, the three LSB of pixel A can also be calculated from pixel C.

4. Experimental results

In this section, we shall show our experimental results and evaluate the performance of our method. To begin with, let us have a look at our test image Lena with 512×512 pixels shown in Fig. 8. We used this test image because there are both detail areas and non-detail areas in it.

First of all, we produced parity check bits for each pixel in the Lena image. Then, the procedure of Torus automorphism was used to decide which pixel to embed the parity check bits in. After that, the parity check bits went through the procedure of bit rotation, and the rotated bits were embedded into the due pixels. Finally, we got the authenticated image under protection from tampering. The authenticated image is shown in Fig. 9.

Our first experiment was meant to show the capability of our proposed method to resist the impact of noise. The authenticated image was taken as a long bit string. In this bit string, we picked up 5000 bits at random and reversed the values of those bits. The resultant partial image is shown in Fig. 10(a). After correcting the pixels by using our proposed method, we attained the resultant partial image, which is shown in Fig. 10(b).

Our second experiment was done to prove the capability of our proposed method to detect the tampered area and recover it. We picked out from the Lena image two areas and tampered them: the first one is a non-detail (smooth) area, and the other is a detail (complex) area. The experimental results of the detail area and non-detail area are shown in Figs. 11 and 12, respectively.

We can see that the experimental results in Figs. 11(c) and (d) are very good. In the picture shown in Fig. 11(c),

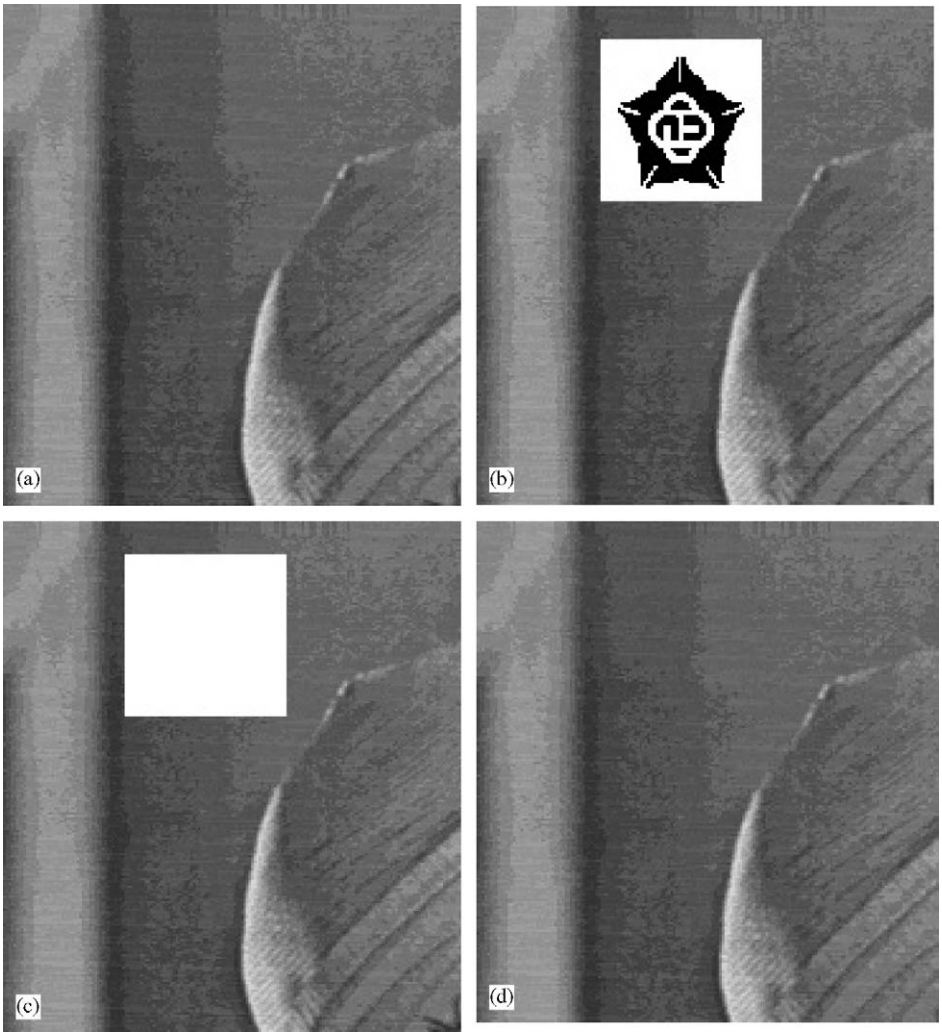

Fig. 14. The resultant partial image for the non-detail area. (a) The partial image of the authenticated image. (b) The partial image with tampered area. (c) The partial image with detected area. (d) The partial image with recovered area.

the tampered area is detected very clearly. Moreover, most of the pixels in the tampered area have been recovered in Fig. 11(d). We can even see the details of the hair strings in the recovered area.

In Fig. 12, we show the non-detail (smooth) area in the resultant images. The partial images are the background of the Lena image.

It is obvious that our proposed method produced good resultant images, which are very similar to the original ones, in our experiments. The tampered area was successfully detected (see Fig. 12(c)), and the area was recovered very well too (see Fig. 12(d)).

In the following experiments, we modified our method a little in order to skip the step of determining the value of the MSB of the data bits mentioned earlier in Section 3.3. Here, we used one bit to record the value of the MSB of the data bits. Then, this bit and the parity check bits were concatenated together and went through the procedure of bit rotation. After that, the rotated bits were embedded in the four LSB of another pixel. The results are shown in Figs. 13 and 14.

We recovered the modified pixels in Fig. 13(d) without any guess in the case where the concatenated data of the modified pixels were not destroyed. The area was recovered without any loss.

The tampered area was also detected (see Fig. 14(c)), and the area was recovered without any loss (see Fig. 14(d)).

5. Conclusions

Many methods have been developed to perform image authentication. However, some problems of those methods, which we mentioned earlier in Section 1, do not seem to have been solved. In this paper, we attempt to reduce some of the disadvantages of those methods. That is the reason why we take the Hamming code as the base on which we develop our method. Taking advantage of the

nature of the Hamming code, we can solve those drawbacks simultaneously.

According to our experimental results, our new method can work quite well in eliminating burst bit errors. Use our method, and the authenticated image will no longer have noise on it. Moreover, no matter whether a detail or non-detail tampered area it is, the recovery can be done very successfully. In a word, our new scheme as a whole is quite a practical method to do tamper proofing.

References

[1] C.C. Chang, C.S. Chan, A watermarking scheme based on principal component analysis technique, Informatica 14 (4) (2003) 431–444.

[2] M.S. Hwang, C.C. Chang, K.F. Hwang, Digital watermarking of images using neural networks, J. Electron. Imaging 9 (4) (2000) 548–555.

[3] Y.W. Kim, I.S. Oh, Watermarking text document images using edge direction histograms, Pattern Recognition Lett. 25 (11) (2004) 1243–1251.

[4] F.Y. Shih, S.Y.T. Wu, Combinational image watermarking in the spatial and frequency domains, Pattern Recognition 36 (4) (2003) 969–975.

[5] P.Y. Tsai, Y.C. Hu, C.C. Chang, A color image watermarking scheme based on color quantization, Signal Process. 84 (4) (2004) 95–106.

[6] Y.T. Wu, Y.F. Shih, An adjusted-purpose digital watermarking technique, Pattern Recognition 37 (12) (2004) 2349–2359.

[7] M.U. Celik, G. Sharmar, A.M. Tekalp, Hierarchical watermarking for secure image authentication with localization, IEEE Trans. Image Process. 11 (6) (2002) 585–594.

[8] C.Y. Lin, S.F. Chang, A robust image authentication method distinguishing JPEG compression from malicious manipulation, IEEE Trans. Circuits Syst. Video Technol. 11 (2) (2001) 153–168.

[9] D.C. Lou, J.L. Liu, Fault resilient and compression tolerant digital signature for image authentication, IEEE Trans. Consumer Electron. 46 (1) (2000) 31–39.

[10] C.S. Lu, H.Y. Liao, Structural digital signature for image authentication: an incidental distortion resistant scheme, IEEE Trans. Multimedia 5 (2) (2003) 161–173.

[11] P.Y. Tsai, Y.C. Hu, C.C. Chang, Using set partitioning in hierarchical trees to authenticate digital image, Signal Process.: Image Commun. J. 18 (2003) 813–822.

[12] P.W. Wong, N. Memon, Secret and public key image watermarking schemes for image authentication and ownership verification, IEEE Trans. Image Process. 10 (10) (2001) 1593–1601.

[13] R.W. Hamming, Error detecting and error correcting codes, Bell Syst. Tech. J. 26 (2) (1950) 147–160.

[14] G. Voyatzis, I. Pitas, Chaotic mixing of digital images and applications to watermarking, Proceedings of European Conference on Multimedia Applications, vol. 2, 1996, pp. 687–695.

[15] J. Serra, Image Analysis and Mathematical Morphology, Academic Press, New York, 1982.