Chapter 7

Data Hiding for Image Authentication

7.1 Introduction

For years, audio, image, and video have played important role in journalism, archiving, and litigation. A coincidentally captured video of the well-known 1993 Rodney King incident played an important role in prosecution, a secretly recorded conversation between Monica Lewinsky and Linda Tripp touched off the 1998 presidential impeachment, just to name a few. Keeping our focus on still pictures, we have no difficulty in realizing that the validity of the old saying "Picture never lies" has been challenged in the digital world of multimedia. Compared with the traditional analog media, making seamless alteration is much easier on digital media by software editing tools. With the popularity of consumer-level scanner, printer, digital camera, and digital camcorder, detecting tampering becomes an important concern. In this chapter, we discuss using digital watermarking techniques to partially solve this problem by embedding authentication data invisibly into digital images.

In general, authenticity is a relative concept: whether an item is authentic or not is relative to a reference or certain type of representation that is regarded as

authentic. Authentication is usually done by checking whether certain rules and relationship which are supposed to be found for an authentic copy are still hold in the test material. In traditional digital network communication, a sophisticated checksum, usually known as hash or message digest, is used to authenticate whether the content has been altered or not. The checksum is encrypted using such cryptographic techniques as public-key encryption to ensure that the checksum cannot be generated or manipulated by unauthorized persons. This is known as *digital signature* in cryptography [11]. For traditional type of source data such as text or executable codes, the checksum is stored or transmitted separately since even minor changes on this kind of data may lead to different meaning. Perceptual data such as digital image, audio and video are different from the traditional digital data in that perceptual data can be slightly changed without introducing noticeable difference, which offers new room for authenticating perceptual data. For example, we can imperceptibly modify an image so that the least significant bit of the representation of each pixel is always the checksum of other bits. In other words, the checksum is *embedded* into the image instead of having to be stored separately as for traditional data. Such embedding approach falls in the category of digital watermarking / data-hiding. For example, *fragile watermarking* [26] can be used to insert into an image some special data which will be altered when the host image is manipulated.

Many general techniques of data hiding can be applied to this specific applications, such as the general approaches discussed in Part I and the image data hiding approaches presented in Chapter 6. But the algorithm design has to be aware of a few unique issues associated with authentication, including the choice of what to embed and the security considerations to prevent forgery or manipulation of embedded data. The following features are desirable to construct an effective authentication scheme:

1. to be able to determine whether an image has been altered or not;
2. to be able to integrate authentication data with host image rather than as a separate data file;
3. the embedded authentication data be invisible under normal viewing conditions;
4. to be able to locate alteration made on the image;
5. to allow the watermarked image be stored in lossy-compression format, or more generally, to distinguish moderate distortion that does not change the high-level content vs. content tampering.

This chapter presents a general framework of watermarking for authentication and proposes a new authentication scheme by embedding a visually meaningful watermark and a set of features in the transform domain of an image via table look-up. The embedding is a Type-II embedding according to Chapter 3. Making use of the pre-distortion nature of Type-II embedding, our proposed approach can be applied to compressed image using JPEG or other compression techniques, and the watermarked image can be kept in the compressed format. The proposed approach therefore allows distinguishing moderate distortion that does not change the high-level content versus content tampering. The alteration made on the marked image can be also localized. These features make the proposed scheme suitable for building a "trustworthy" digital camera. We also demonstrate the use of shuffling (Chapter 4) in this specific problem to equalize uneven embedding capacity and to enhance both embedding rate and security.

7.2 Previous Work

The existing works on image authentication can be classified into several categories: digital signature based, pixel-domain embedding, and transform-domain embedding. The latter two categories are invisible fragile or semi-fragile watermarking.

Digital signature schemes are built on the ideas of hash (or called message digest) and public key encryption that were originally developed for verifying the authenticity of text data in network communication. Friedman [74] extended it to digital image as follows. A signature computed from the image data is stored separately for future verification. This image signature can be regarded as a special encrypted checksum. It is unlikely that two different natural images have same signature, and even if a single bit of image data changes, the signature may be totally different. Furthermore, public-key encryption makes it very difficult to forge signature, ensuring a high security level. After his work, Schneider *et al.* [75] and Storck [79] proposed content-based signature. They produce signatures from low-level content features, such as block mean intensity, to protect image content instead of the exact representation. Another content-signature approach by Lin *et al.* developed the signature based on a relation between coefficient pairs that is invariant before and after compression [41, 76]. Strictly speaking, these signature schemes do not belong to watermarking since the signature is stored separately instead of embedding into images.

Several pixel-domain embedding approaches have been proposed. In Yeung *et al.*'s work, a meaningful binary pattern is embedded by enforcing certain relationship according to a look-up table. Their work allows the tampering that is confined in some local areas to be located [78]. Walton proposed an approach by embedding data via enforcing relationship between sets of pixels [77]. Another pixel-domain scheme

was proposed by Wong [80]. This scheme divides an image into blocks, then copies the cryptographic digital signature of each block in the least significant bits of the pixels for future verification. However, images marked with these pixel-domain *Invisible-Fragile Watermarking* schemes cannot be stored in lossily compressed format like JPEG compression, which is commonly used in commercial digital camera.

In addition to pixel-domain approaches, several block DCT-domain schemes may be used for authentication purpose. Swanson *et al.* [66] round coefficients to multiples of just-noticeable difference or mask value, then add or subtract a quarter to embed one bit in an 8×8 block. Koch *et al* [63] embed one bit by forcing relationships on a coefficient pair or triplet in mid-band. The two approaches achieve limited robustness via pre-distortion, and the embedding is likely to introduce artifacts in smooth regions. Similar problem exists in other approaches, including a DCT-domain quantization approach by Lin *et al.* [41], and a Wavelet-domain quantization approach by Kundur *et al.* [82], both of which embed a signature in transform domain by rounding the quantized coefficients to an odd or even number. Additional data can be embedded to recover some tampered or corrupted regions, such as the self-recovery watermarking proposed by Fridrich *et al.* in [88] and by Lin *et al.* in [41].

Recalling the desirable requirement for image authentication presented in the previous section, we find that many existing approaches in literature cannot satisfy all requirements. The digital signature proposed in[74], as well as the content based signature reported in [75] and [79] do not satisfy the requirements 2 and 4. The pixel-domain scheme [78, 77, 80] can not be stored in lossy compression format. In addition, transform-domain schemes [41, 82, 63, 66] do not handle the uneven embedding capacity problem raised in Chapter 4, therefore may either introduce artifacts in smooth region or embed fewer authentication bits than our proposed approach.

7.3 Framework for Authentication Watermark

We proposed a general framework including the following elements for authentication watermark:

1. what to authenticate
2. what data to be embedded for authentication purpose
3. how to embed data into an image
4. how to handle uneven embedding capacity
5. security considerations

The first element is fundamental and affects the other elements. We have to decide whether to authenticate the exact representation of an image, or to have some tolerance toward certain processing such as compression, cropping and scaling. In addition, we need to determine other functionalities we would like to achieve, such as locating alteration. The next two elements, namely, what and how to embed, are designed based on the answer to the first element. More specifically, we can either mainly rely on the fragility of the embedding mechanism to detect tampering (e.g., to put zeros in the least significant bits of all pixel values and later to check whether such properties still hold or not on test images), or rely on the embedded data (e.g., to robustly embed a 64-bit check sum of the image features such as the block mean intensity or image edge map, and later to check whether the extracted check sum matches the one computed from the test image), or use both. For local embedding schemes such as the TDMA type modulation discussed in Chapter 4, special handling with smooth region, or in general, the uneven embedding capacity is necessary to achieve high embedding rate and to locate alteration. Besides an appropriate design

of what and how to embed, the detailed implementation must take security issues into account in order to meet the demands in practical applications, for example, to make it difficult for an attacker to forge valid authentication watermark in a tampered image.

Following the above framework, we discuss our proposed authentication watermarking approach based both on the fragility of embedding mechanism and on matching the embedded features with the features extracted from a test image. The detection of tampering relies on both the embedding mechanism and the embedded data. The alteration can also be located unless there is global tampering or the tampering involves large areas. We shall present our approach in the context of grayscale images with JPEG compression. The extension to images compressed using other means such as Wavelet and to color images will be briefly discussed in Section 7.7. A block diagram of the embedding process is given in Fig. 7.1 which, aside from the block labeled "embed", is identical to that of the JPEG compression [18]. Watermarks are inserted into the quantized DCT coefficients via a look-up table. Explained below are two aspects of watermark-based authentication, namely, to embed what data and how to embed them.

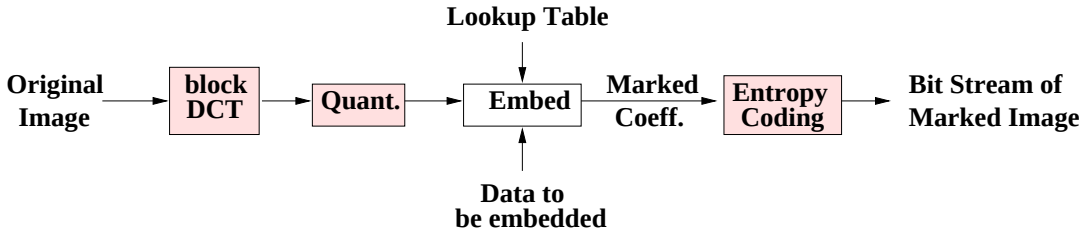

Figure 7.1: Block diagram of embedding process for authentication watermarking. The "Quant." module stands for the quantization step.

7.4 Transform-domain Table Lookup Embedding

The data for authentication purpose is generally embedded using a Type-II approach discussed in Chapter 3 for its high embedding capacity and fragility, both of which are useful in authentication. Here we present a Type-II embedding using a look-up table in transformed domain. This transform domain look-up table embedding is an extension of the pixel-domain scheme proposed by Yeung *et al.* [78]. The embedding is performed on the quantized coefficients with a set of pre-selected quantization step sizes, which are known to the detector because the extraction of hidden data must be performed in the same quantized domain. As discussed in Chapter 3, this quantization is a pre-distortion step to obtain limited robustness against compression and other distortion.

A proprietary look-up table (LUT) is generated beforehand by the owner of the image or the digital camera manufacturers. The table maps every possible value of JPEG coefficient randomly to "1" or "0" with a constraint that the runs of "1" and "0" are limited in length. To embed a "1" in a coefficient, the coefficient is unchanged if the entry of the table corresponding to that coefficient is also a "1". If the entry of the table is a "0", then the coefficient is changed to its nearest neighboring values for which the entry is "1". The embedding of a "0" is similar. This process can be abstracted into the following formula where v_i is the original coefficient, v_i' is the marked one, b_i is the bit to be embedded in, $Q(\cdot)$ is the quantization operation [1], and

[1]For a uniform quantizer with quantization step size q, the quantization operation $Q(x)$ is to round x to the nearest integer multiples of q.

$LUT[\cdot]$ is the mapping by a look-up table:

$$v_i{}' = \begin{cases} Q(v_i) & \text{if } LUT[Q(v_i)] = b_i \\ v_i + \delta & \text{if } LUT[Q(v_i)] \neq b_i, \text{ and} \\ & \quad \delta = \min_{|d|}\{d = Q(x) - v_i \text{ s.t. } LUT[Q(x)] = b_i\} \end{cases} \tag{7.1}$$

The extraction of the signature is simply by looking up the table. That is,

$$\hat{b_i} = LUT[Q(v_i{}')] \tag{7.2}$$

where $\hat{b_i}$ is the extracted bit.

The basic idea of the embedding process is also illustrated by the example in Fig. 7.2. Here, zeros are to be embedded in two quantized AC coefficients with values "-73" and "-24" of an 8×8 image block. The entry in the table for coefficient value "-73" is "1". In order to embed a "0", we have to change it to its closest neighbor for which the entry is "0". In this example, "-73" is changed to "-74". Since the entry for coefficient value "24" is already "0", it is unchanged.

As mentioned earlier, the detection of tampering is based on both the embedding mechanism and the embedded data. The clues provided by the embedding mechanism is as follows: when a small part of a watermarked image is tampered without the knowledge of the look-up table, the extracted bit from each tampered coefficient becomes random, i.e.,

$$P(\hat{b_i} = 0) = P(\hat{b_i} = 1) = \frac{1}{2}$$

implying that it is equally likely to be the same as or be different from the bit b_i originally embedded. For the moment, we assume that detector has knowledge of the originally embedded data $\{b_i\}$ and the justification of this assumption will be presented later. From a single coefficient, it is not reliable to determine whether tampering occurs or not because there is a 50/50 chance that the extracted data

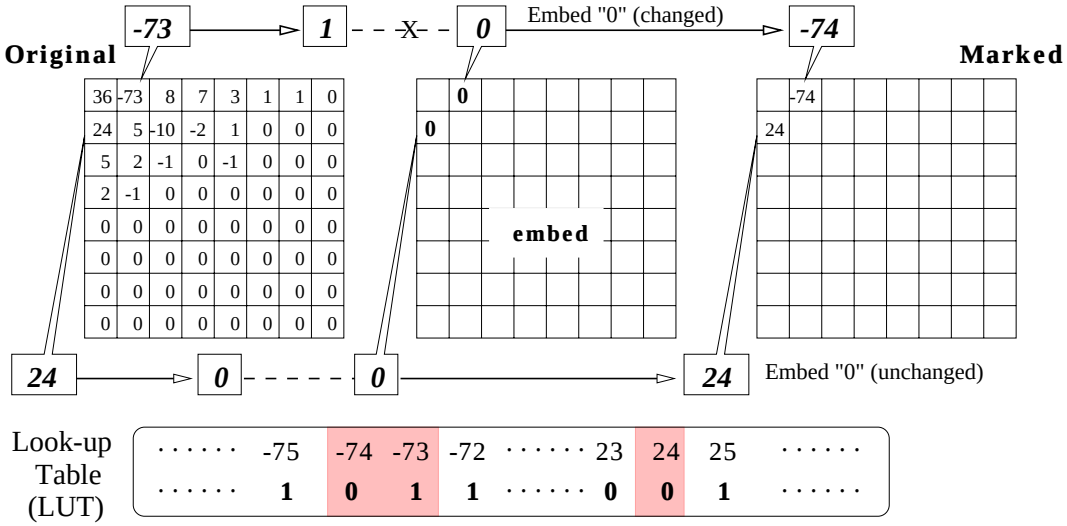

Figure 7.2: Frequency-domain Embedding Via Table Lookup: zeros are embedded to two quantized DCT coefficients "-73" and "24" by enforcing relationship according to a look-up table. After embedding, the coefficients in the watermarked image are "-74" and "24".

matches the originally embedded one, i.e., $\hat{b_i} = b_i$. However, if the tampering affects several coefficients in a block and/or coefficients in several blocks, the chances of miss detection (i.e., all decoded data of altered region happen to match the originally embedded ones) are reduced exponentially:

$$P([\hat{b_1}, ..., \hat{b}_n] = [b_1, ..., b_n]) = 2^{-n}$$

where n is the number of coefficients affected by tampering. For example, miss detection probability is around 0.00098 when n is equal to 10.

According to Chapter 3, the table lookup embedding, being an example of Type-II embedding, relies on deterministic relationship enforcement. From set partition point of view, all possible values of a quantized coefficient are divided into two subsets each of which conveys special meaning and the partition rule is set by the table. One

subset contains values which map to "1" according to the lookup table, and the other subset contains those that map to "0". The embedding process introduces minimal necessary changes to force a quantized coefficient to take value from the subset that maps to the binary data to be embedded.

7.4.1 Considerations for Imperceptibility & Security

Several steps are taken to ensure that the embedding is invisible:

- The run of "1" and "0" entries in the LUT is constrained to avoid excessive modification on the coefficients;
- The DC coefficient in each block is not changed to avoid blocky effect unless the quantization step is very small [2];
- Small valued coefficients (mostly in high frequency bands) are not modified to avoid large relative distortion.

Coefficients that are allowed to be changed according to these constraints are called *embeddable* or *changeable*. The number embeddable coefficients vary significantly, which has been referred as "uneven embedding capacity" in Chapter 4. Also, extraction error may occur due to image format conversion, rounding, and other causes involving no content changes. As discussed in Chapter 4, we apply shuffling to equalize the unevenly distributed embedding capacity. A proper block size is first determined according to the overall embedding capacity measured by the total number of changeable coefficients. The side information regarding the block size can be

[2]This constraint may be loosened to allow the DC coefficients in texture regions to be modified, as the change there is likely to be invisible.

conveyed using the approaches discussed in Chapter 4 and 6, for example, using additive spread spectrum embedding. Then, one bit is embedded in each shuffled block by repeatedly embedding the same bit to all embeddable coefficients in the shuffled block. The bit is extracted by a detector in the same shuffled domain via majority voting. The repetition may be replaced by more sophisticated coding techniques for better performance.

The algorithm shown in Table 7.1 is for generating an L-entry look-up table $T[\cdot]$ with maximum allowed run of r and positive index $i \in \{1, ..., L\}$.

Table 7.1: An Algorithm for Generating Look-up Table with Limited Runs

Step-1:	$i = 1$.
Step-2:	If $i > r$ and $T[i-1] = T[i-2] = ... = T[i-r]$, then $T[i] = 1 - T[i-1]$. Otherwise, generate a random number from $\{0, 1\}$ with a probability $0.5 : 0.5$ and set $T[i]$ to this value.
Step-3:	Increase i by 1. If $i > L$, stop. Otherwise go back to Step-2.

To analyze the minimum secure value of r, we start with the case of $r = 1$, which has only two possibilities:

$$T[i+1] = 1 - T[i], \qquad T[0] \in \{0, 1\}, i \in \mathbf{N}$$

or equivalently,

$$T[i] = \begin{cases} 0 & \text{(i is even)} \\ 1 & \text{(i is odd)} \end{cases} \qquad \textit{or} \qquad T[i] = \begin{cases} 1 & \text{(i is even)} \\ 0 & \text{(i is odd)} \end{cases}$$

This implies that the odd-even embedding discussed in Chapter 3 is a special case of table-lookup embedding. Since there is very little uncertainty in the table, it is easy for unauthorized persons to manipulate the embedded data, and/or to change the coefficient values while retaining the embedded data unchanged. Therefore, $r = 1$ is

not a proper choice if no other security measure such as a careful design of what data to embed is taken.

When r is increased to 2, the transition of the LUT entries has Markovian property, as shown in Fig. 7.3. It is easy to show that starting from "0" or "1", the number of possible LUTs with i elements long, F_i, forms a *Fibonacci series*:

$$F_i = F_{i-1} + F_{i-2}, \qquad F_0 = 1, F_1 = 1. \tag{7.3}$$

The total number of possible sequences with length $L = 256$ is on the order of 10^{53}. Although this number is smaller than the number of possible sequences without run length constraint, which is 2^{256}, or the order of 10^{77}, the table still has enough uncertainty, and the probability of obtaining table by guessing is very, very small. Thus from embedding mechanism point of view, the minimum secure choice for r is 2.

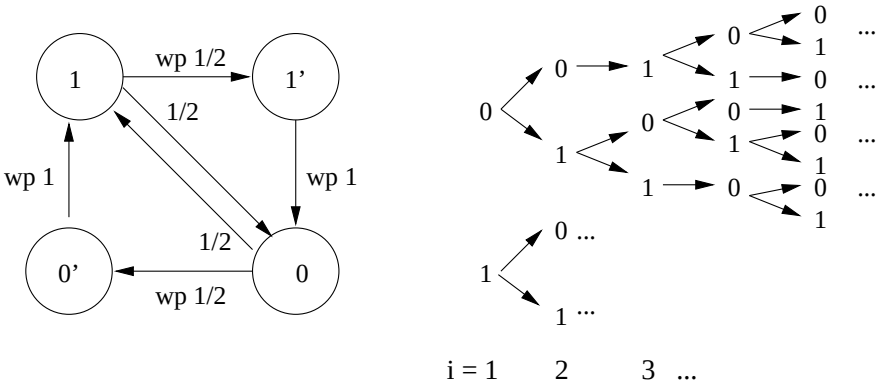

Figure 7.3: (left) Markov property of restricted LUT generation with maximum run of 2, where "wp" stands for "with probability"; (right) Expansion graph illustrating the generation of restricted LUTs with i elements.

The mean squared error incurred by table-lookup embedding with $r = 2$ is computed as follows. First, we consider the error incurred purely by quantization, i.e.,

rounding the input coefficient in the range of $[(k-1/2)Q, (k+1/2)Q)$ to kQ:

$$\text{MSE(quantize to } kQ \text{)} \quad = \quad \frac{Q^2}{12} \tag{7.4}$$

This is the case if the entry corresponding to the original quantized coefficient in the table happens to be the same as the bit to be embedded. We then consider the case of having to shift the coefficient to $(k-1)Q$ in order to embed the desired bit:

$$\begin{aligned} \text{MSE(shift to } (k-1)Q \text{)} &= \int_{(k-\frac{1}{2})Q}^{(k+\frac{1}{2})Q} [x-(k-1)Q]^2 \times \frac{1}{Q} dx \\ &= \underset{\text{let} \quad y = x-(k-1)Q}{\int_{\frac{Q}{2}}^{\frac{3}{2}Q}} \frac{1}{Q} \times y^2 dy \quad = \quad \frac{13}{12}Q^2 \end{aligned} \tag{7.5}$$

By symmetry, the MSE for shifting to $(k+1)Q$ is same as above. Hence the overall MSE is:

$$\begin{aligned} \text{overall MSE} &= \frac{\text{MSE(to kQ)}}{2} + \frac{\text{MSE(to (k-1)Q)} \; + \; \text{MSE(to (k+1)Q)}}{4} \\ &= \frac{1}{2} \times \frac{Q^2}{12} + 2 \times \frac{1}{4} \times \frac{13}{12}Q^2 = \frac{7}{12}Q^2 \end{aligned} \tag{7.6}$$

This is achievable since look-up table with $r = 2$ allows at most one-Q-step modification away from kQ. The MSE is a little larger than the case of $r = 1$ (i.e., the odd-even embedding in Chapter 3), which gives an MSE of $Q^2/3$ and is equivalent to double the quantization size as far as the distortion is concerned.

7.4.2 More on Determining Embedded Data & Coefficient Changes

Earlier when explaining the extraction of embedded data, we have assumed that a detector has the knowledge of the originally embedded data $\{b_i\}$. In practice, this

knowledge is not a necessity, especially with the incorporation of majority voting, error correction coding and shuffling.

Assuming that the tampering is restricted to a small portion of an image, the changed coefficients tend to occur in small clusters and the number of them is not large. After shuffling, these coefficients will be diffused to many blocks in shuffled domain. Because each shuffled block is unlikely to receive many too many changed coefficients by the nature of shuffling [3], the few changed coefficients in each shuffled block will not affect $\{\hat{b}_i\}$, which is the bit ultimately extracted from that block via table lookup detection and error correction coding. This implies that the extracted data, $\{\hat{b}_i\}$, can be regarded as a good estimate of $\{b_i\}$, the originally embedded data. Using $\{\hat{b}_i\}$ as "ground truth", we can determine what bit value is supposed to be embedded into each embeddable coefficients by an embedding system. By comparing the supposedly embedded data with the data actually extracted from each coefficients of a test image, we are able to identify the changed coefficients.

The change identification by this two-step process relies on the fragility of embedding, namely, the tampering may change the originally embedded data. In the next section, we shall see a second way to detect tampering, which relies on the meaning and the value of embedded data. By then, we will get a more complete picture of the authentication framework introduced in Section 7.3.

7.5 Design of Embedded Data

We mentioned earlier that the embedded data can be used to detect tampering. The authentication data we propose to embed consists of a visually meaningful binary

[3] This is supported by the same analysis on shuffling in Section 4.7, with the percentage of blue balls (the balls of interest), p, being very small.

pattern and content features. As we shall see, the combination of these two types data is suitable for applications such as image authentication for "trustworthy" digital cameras.

7.5.1 Visually Meaningful Pattern

The visually meaningful pattern, such as letters and logos, serves as a quick check for signaling and locating tampering. Shown in Fig. 7.4 is a binary pattern "PUEE". The decision on whether an image is altered or not can be made by (automatically) comparing the extracted pattern with the original one, if available, or by human (e.g., jury in court) based on visualizing the extracted pattern. The latter case uses a reasonable assumption that human can distinguish a "meaningful" pattern from a random one. It is also possible to make such decision automatically, e.g., computing a randomness measure.

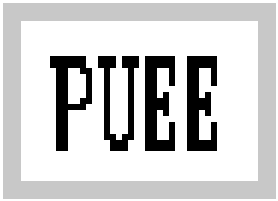

Figure 7.4: A binary pattern as part of the embedded data

7.5.2 Content-based Features

Content features offer additional security to combat forgery attack and help distinguish content tampering vs. minor, non-content change. Due to the limited embedding bit rate, content features have to be represented using very few number of bits.

An example of content features is the most significant bit of macroblock mean intensity. Another example is the sign of intensity difference between a pair of blocks or macroblocks. These features bring dependence on images to the data to be embedded, so it is effective against forgery attacks that rely on the independence [139]. Other features, such as the edge map and luminance/color histogram, can also be used. A general formulation of local features of the block (i, j) is

$$b_{i,j} = f(\underline{v}_{i-k_1,j-k_2}, ..., \underline{v}_{i-1,j}, \underline{v}_{i,j}, \underline{v}_{i+1,j}, ..., \underline{v}_{i+k_3,j+k_4})$$

where $k_1, k_2, k_3, k_4 \in \mathbf{N}$, $\underline{v}_{i,j}$ represents a collection of all the coefficients in the block (i, j), and $f(\cdot)$ is a deterministic function which is known to both the embedder and the detector. The detector compares the features computed from the test image and the ones extracted by table look-up (i.e., the features embedded by the embedder). A mis-match between the two sets of features is an indication that the image has been tampered.

Content features are especially useful in authenticating smooth regions. As discussed in Section 7.4.1, no data can be embedded in smooth regions without introducing visible artifacts, hence it is impossible to rely on the embedding mechanism to signal the tampering of these regions, i.e., it is impossible to embed data with certain regularity and later check the alteration of such regularity at detector. The tampering associated with smooth regions includes the case of altering a smooth block to another smooth one (e.g., changing luminance or color) and the case of altering a complex block to a smooth one. Changing smooth block to complex block is easy to detect because when the detector sees the complex block, it assumes some data have been embedded, but the extracted data from these altered block will be random as an attacker has no knowledge of the secrets used in embedding. Although there are limited meaningful changes that can be applied by changing original blocks (either

smooth or complex) to smooth ones, we believe that authentication schemes should take smooth region authentication into account instead of risking miss detection of possible meaningful alterations. Since the embedded data can be used to detect tampering, alterations in smooth regions can be detected by relying on the embedded data, i.e., we embed features derived from smooth regions in the embeddable regions and later check the match between the features computed from a test image and those extracted by the watermark detection module. In addition, the features based on block mean intensity are useful in detecting intentional alterations such as the possible meaningful tampering by only changing some DC coefficients, because the embedding scheme preferably leaves DC coefficient untouched to avoid blocky effect.

7.6 Experimental Results

We first present the results of an earlier design that does not use shuffling and embeds less data than the shuffling approach. A JPEG compressed image of size 640×432 is shown in Fig. 7.5. Fig. 7.6(a) is the same image but with a 40×27 binary pattern "PUEE" of Fig. 7.6(b) and the MSB of macroblock mean intensity embedded in. This image is visually indistinguishable from the original. In terms of PSNR with respect to original uncompressed image, the watermarked one is only 1dB inferior to the image with JPEG compression. The smooth regions of the image are mainly in the sky area. For these blocks, *backup embedding* is used, namely, the data associated with the i-th block are embedded in both the i-th block and a companion block (see Fig. 4.4).

The marked image is modified by changing "Princeton University" to "Alexander Hall" and "(c) Copyright 1997" to "January 1998", shown in Fig. 7.7(a). This image

Figure 7.5: An original unmarked 640×432 image *Alexander Hall* (stored in JPEG format with a quality factor of 75%). Watermarking is performed on its luminance components.

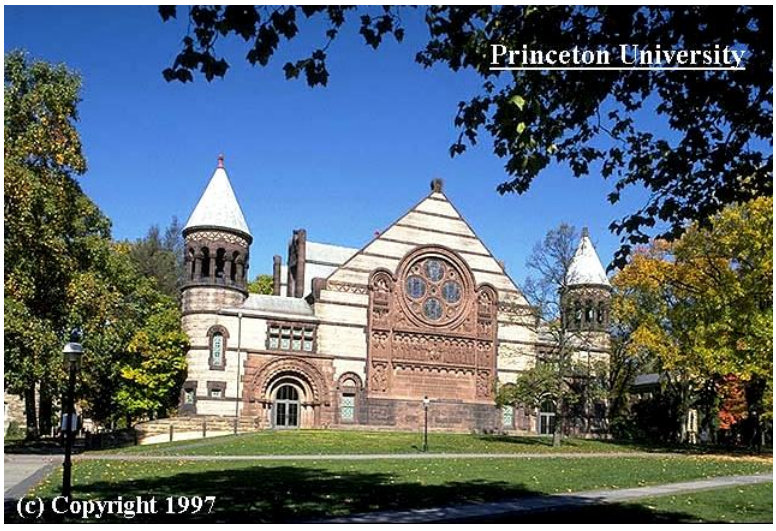

Figure 7.6: $[a|b]$ Authentication result with watermark embedded without shuffling: (a) watermarked image; (b) embedded binary pattern.

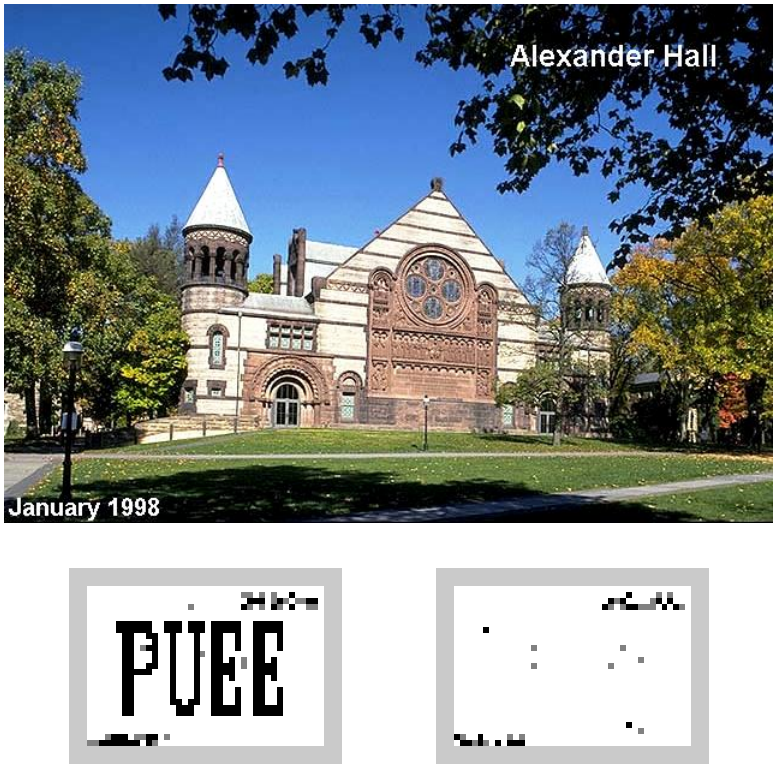

Figure 7.7: $[\frac{a}{b|c}]$ Authentication result with watermark embedded without shuffling: (a) an altered version of the watermarked image (stored in 75% JPEG); (b) extracted binary pattern from edited image; (c) feature matching result.

is again stored in the JPEG format. For the ease of visualizing embedded pattern, we shall denote the block that embeds "0" as a black dot, the block that embeds "1" as a white dot, and the block with no obvious majority being found in detection as a gray dot. Similarly, to visualize the feature matching result, we use a black dot for the unmatched block, a white for the matched one, and a gray for the block with which we do not found the obvious majority in detection hence have low confidence in determining a match or unmatch. With these notations, Fig. 7.7(b) and (c) show the extracted binary pattern and content feature matching result of the tampered image.

The random pattern corresponding to the tampered blocks are clearly identifiable. We can see that using the backup embedding, there are very few unembeddable bits at an expense of the reduced embedding rate. Also notice that the round-off error may occur during the verification and tampering, which contributes to the few unexpected dots out of the altered regions.

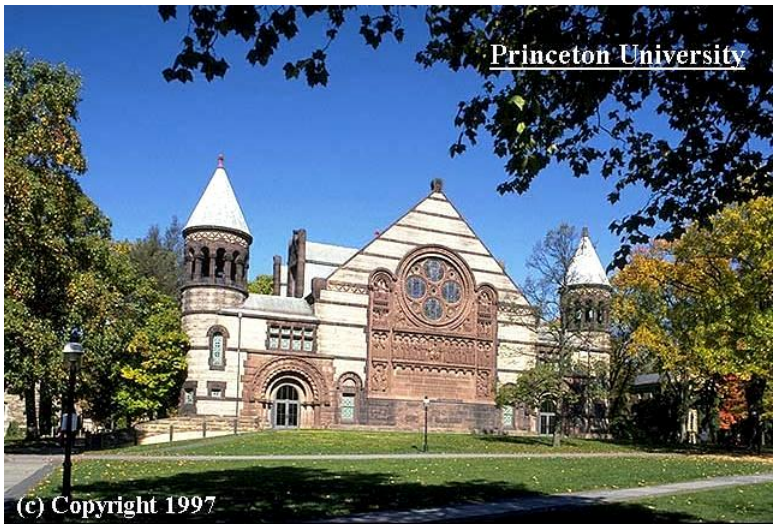

Figure 7.8: An image with authentication watermark embedded using shuffling.

As discussed in Chapter 4, shuffling equalizes the uneven embedding capacity and allows embedding more data. The example shown in Fig. 7.8 has a BCH encoded version of the "PUEE" pattern and multiple content features embedded in its luminance components. There is no visual difference between this watermarked image and the original unwatermarked copy in Fig. 7.5. The embedded content features include the most significant bit of the average macroblock intensity and the smoothness of each macroblock. The combined result of pattern comparison and feature matching provides information regarding both content tampering and minor non-content changes

(a)

(b)

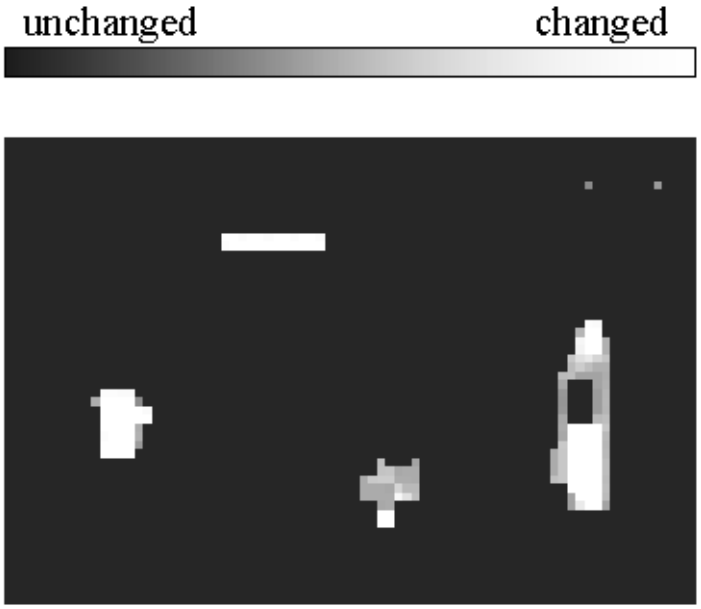

Figure 7.9: Authentication result with watermark embedded using shuffling: (a) an altered version stored in 75% JPEG format of Fig. 7.8; (b) authentication result with whiter dot denoting higher likelihood of content manipulation for the corresponding area in (a).

such as those introduced by rounding or recompression. Fig. 7.9(a) shows various content alterations made to the watermarked image. A comprehensive report combining both pattern comparison and feature matching result is shown in Fig. 7.9(b), with whiter pixel indicates higher likelihood of content changes. We can see that the improved authentication system using shuffling is able to identify content changes with little false alarm. The quantization domain used by the embedding step is the same as JPEG with quality factor 75%, implying the system is able to tolerate compressions and other distortions that are comparable or less severe than JPEG 75%.

7.7 Extensions

Multi-level Data Embedding and Unequal Error Protection

We mentioned in Section 7.5 that two sets of data, namely, a meaningful visual pattern and a set of low-level content features, are embedded in the image for authentication purpose. More generally, the idea of multilevel data hiding in Chapter 6 can be incorporated to embed several sets of data with different levels of error protection and using embedding mechanisms with different robustness. The embedded data sets could be image features at different resolutions: the coarser the level is, the more it is protected. The multi-resolution features with unequal error protection can help us authenticate an image in a hierarchical way.

Other Compression Format

Besides JPEG compressed image, we also performed some simple tests in Wavelet compression and found our approach effective in detecting tampering. In addition to the advantage in efficient coding, Wavelet domain offers convenience to implement the above-mentioned hierarchical authentication. Since wavelet is selected as the core of

the new JPEG2000 compression standard, a complete authentication system design targeted at JPEG2000 is a promising new direction.

Color Images

For color images, we currently work in YCrCb coordinates and use the proposed approach to mark luminance components while leaving chrominance components unchanged. A better way is to apply the approach to chrominance components as well to embed more data and to detect potential tampering of colors. We may also work in other color coordinates, such as RGB. However in practice, due to the limited computation precision, we are likely to find a few pixels whose YCrCb or RGB values may change after one or more rounds of color coordinate conversion [152]. This is similar to the rounding and truncating errors incurred by going to and forth between pixel domain and transform domain. Pre-distortion via quantization and error correction codes are ways to combat these errors.

Videos

A system for authenticating MPEG compressed digital video can be designed by marking I-frames of video streams using our proposed scheme. In addition, I-frame serial number can be used as part of embedded data to detect modification such as frame reordering and frame dropping. Alternatively, these frame jitters can be detected via a spread-spectrum watermark in each frame, which is embedded using the same approach as the embedding of frame synch index in Section 6.4. P- and B-frames can be similarly watermarked but with larger quantization step size and more error protection in embedding to survive motion compensation during moderate compression. This practical consideration is similar our implementation of multilevel data hiding for video in Section 6.4. Compressed domain embedding in

these predicted P- and B- frames are also possible by manipulating the residues of motion compensation.