July 2, 2001
RT0419
Rights Management 6 pages

Research Report

A Method to Detect Alterations of Still Images using Quantization Step Characteristics

Kohichi Kamijoh

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

A Method to Detect Alterations of Still Images using Quantization Step Characteristics

Koichi Kamijo
IBM Research, Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken, 242-8502, Japan
kamijoh@jp.ibm.com

Abstract

Insurance companies must detect falsified claims which sometimes include modified photographs. One way to detect such doctored images is to combine device authentication with image authentication. In this paper, we focus on image authentication and report a new alteration detection method using quantization step characteristics in a compressed domain. We also describe results of experiments involving embedding and detection, and the processing time required.
keywords*:* watermarking, Q step, alteration detection, DCT, JPEG, overflow, underflow

1 Introduction

In [1], we described the steps of the transactions involved in filling an insurance claim for a loss, such as taking a picture for evidence using a digital camera, uploading it to a PC, and adding authentication marks for device and image authentication, and we introduced a prototype system named SIMON. Device authentication is a method to guarantee the integrity of the data by authentication between devices by using a protocol prior to transferring the data, and in the case of SIMON, the pictures are taken by the digital camera and uploaded to PCs corresponded to the data. On the other hand, image authentication is a method to guarantee the integrity of the data by embedding an authentication mark into the image itself by using such technologies as DataHiding.

There are several standard image authentication methods, such as calculating the hash of some characteristic values of the image, and embedding some authentication marks using DCT coefficients in the JPEG domain, but those methods have the problem that the authentication marks are altered by format transformations, especially by overflow or underflow routines when the compressed image is decoded, and we cannot tell if the alteration is caused by a simple format transformation or by intentional alteration.

In this paper, we discuss an alteration detection method that solves this problem by using DataHiding, making use of the characteristics of the quantization step (hereafter, Q step). In Section **2**, we discuss a typical classic alteration detection method, in Section **3**, we discuss our method, in Section **4**, we show the experimental results and their performance, in Section **5**, we have a discussion of theoretic reliability, in Section **6**, we briefly discuss the application of our method, and in the final section, we offer our conclusions.

2 Typical classical alteration detection method

As a typical alteration detection method, we calculate the hash value E from the target image I using a hash function H and a key K as

$$E = H(K, I), \tag{1}$$

and then attach E to the header of the image or embed E into the image itself using a technology such as DataHiding, and finally we compare the hash value calculated from target image I', which is to be verified, and the value E, to detect if it has been altered or not [2].

The problem with this method is that the hash value is changed even by such a slight change as a format transformation, so we cannot tell if the mismatch of the hash value is caused by a format transformation or by an intentional alteration, and it leads to false positive errors (hereafter, FP) which mis-judges that the content or the image has been intentionally altered when it has actually not been touched.

3 An alteration detection method making use of the characteristics of the quantization step

In this paper, in order to solve this problem, we introduce an alteration detection method to embed the authentication mark into the image using DataHiding, calculate the hash value of the image, and embed the value into the image itself, making use of the characteristics of the quantization step. We assume that the embedder and the detector share the following information:

1. The DCT indexes $\{d_l\}(0 < d_l < 64)$ used for the embedding and the detection
2. The hash function used for the embedding and the detection
3. The secret key used for the embedding and the detection
4. The area $\boldsymbol{A}$ in the image to be used to embed the hash value

Our method supports such formats as BMP, YUV, JPEG, etc. for the embedding and the detection, and the format at the embedding and the detection times need not be the same, but, to simplify the discussion, we discuss only the case of JPEG (4:1:1) and BMP. Also, we use luminance values for the embedding and the detection.

3.1 Embedding

The embedding procedure consists of "Pre-embedding" (JPEG decoding, BMP→JPG' transformation, Q_e calculation), "Authentication mark embedding", "Post-embedding" (JPG'→BMP transformation, BMP→JPG' transformation, DCT coefficients verification), and "Hash value embedding" (Fig. 1), where "JPG'" is a JPEG format whose DCT coefficients are all de-quantized. We explain the embedding procedure based on Fig. 1.

JPEG decoding: If the input image I is in JPEG format, we Huffman decode, de-quantize all the luminance and color coefficients, and transfer each 8×8 pixel block to an image matrix $\boldsymbol{Y}_s = [\boldsymbol{YUV}]$. Here, each $\boldsymbol{Y}$,$\boldsymbol{U}$,$\boldsymbol{V}$ is a matrix which re-orders the Y, U, V components of each 8×8 DCT block to a 64×1 vector, and $\boldsymbol{Y}_s$ is a 64×3 matrix.

BMP→ JPG' transformation-1: If the input image I is in BMP format, we perform a BMP→ JPG' transformation, transferring all the blocks in I to JPG' format matrices $\boldsymbol{Y}_s$.

Qe calculation: We calculate the embedding Q value 'Qe_l' for the luminance DCT index d_l, which is used for the embedding. This is the Q value used only for the embedding, and we don't have to modify the Q table of the JPEG image. Now, we define δ as twice of the value of the maximum calculation error of the iDCT calculation using systems (decoders), Q_l as the Q value which corresponds to d_l when I is in the JPEG format, and Qe_l as

$$Qe_l = \begin{cases} [(\delta - 1)/Q_l + 1]Q_l, & I\text{: JPG} \\ \delta, & I\text{: BMP.} \end{cases} \tag{2}$$

We define Qe_l to be larger than δ in order to avoid a detection error caused by the format transformation after embedding, and to be an integer multiple value of Q_l so that we need not change the Q table of the original image.

Authentication mark embedding: We set the coefficients of d_l to be the integer multiple of Qe_l so that the quantized values are not changed after JPG'→BMP transformation. First, we define $c(i, j, k)$ as the luminance DCT coefficient of the i-th block, and DCT index k $(0 < k < 64)$ where j=0,1,2 corresponds to the Y,U,V factors, respectively. Therefore, in the case of a 640×480 pixel image, i=0,..,4799.
Next, we modify $c(i, 0, d_l)$ so that it satisfies

$$c(i, 0, d_l) = nQe_l, \quad n = 0, \pm 1, \pm 2, ... \tag{3}$$

for all the i and l, and finally produce JPG' image matrices $\boldsymbol{Y}_{s2}$.

JPG'→BMP transformation: We perform JPG'→BMP transformation, transforming each block

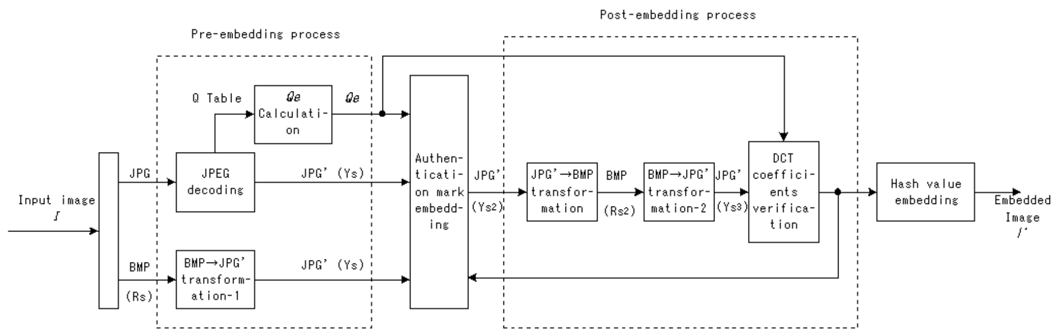

Figure 1. Embedding procedure

of I to BMP format image matrices $\boldsymbol{R}_{s2}$. We now define $\boldsymbol{D}$ as a 64×64 matrix which performs the DCT transformation, and $\boldsymbol{B}$ as a 3×3 matrix which performs the YUV to BMP transformation, and now $\boldsymbol{R}_{s2}$ can be written as

$$\boldsymbol{R}_{s2} = Tr(\boldsymbol{D}^{-1}\boldsymbol{Y}_{s2}\boldsymbol{B}) \tag{4}$$

$$Tr(x) = \begin{cases} T_u, & x > T_u \\ T_d, & x < T_d \\ x, & \text{otherwise} \end{cases}$$

$$T_u, T_d : \text{rounding values, } T_u > T_d.$$

Here, the non-linear function Tr is the cause of FP. One idea to avoid FP is that we choose the embedding vector $\Delta\boldsymbol{Y}_s$ to satisfy

$$T_d\boldsymbol{U} \leq \boldsymbol{D}^{-1}(\boldsymbol{Y}_s + \Delta\boldsymbol{Y}_s)\boldsymbol{B} \leq T_u\boldsymbol{U} \tag{5}$$

where $\boldsymbol{Y}_{s2} = \boldsymbol{Y}_s + \Delta\boldsymbol{Y}_s$ and $\boldsymbol{U}$ is a 64×3 matrix whose elemenents are all "1". But in that case, Eq. (5) becomes a set of n_dth 384 dimensional simultaneous inequality equations, and we may not have any solution for $\Delta\boldsymbol{Y}_s$, because many JPEG images tend to have many overflows or underflows when transformed to BMP (e.g. Table 1), and in that case, we have many blocks which don't satisfy Eq. (5) prior to the embedding ($\Delta\boldsymbol{Y}_s = 0\boldsymbol{U}$).

Our method loops between "Post-embedding" and "Authentication mark embedding" until all the DCT blocks reach the stable condition, which is described later, so that the embedded image doesn't cause FPs.

BMP→JPG' transformation-2: We perform a BMP→JPG' transformation on $\boldsymbol{R}_{s2}$, and create a JPG' format image $\boldsymbol{Y}_{s3}$.

DCT coefficients verification: We verify if $c(i,0,d_l)$, the coefficient of DCT luminance index d_l of $\boldsymbol{Y}_{s3}$, is close to an integer multiple of Qe_l. In other words, we verify if

$$|nQe_l - c(i,0,d_l)| < \epsilon \text{ for } \exists n \tag{6}$$
$$n = 0, \pm 1, \pm 2..$$

is satisfied for all the i and l for a threshold ϵ. When a block i satisfies Eq. (6) for all the values of l, we call the block "stable", and the embedding into the block is completed. For the blocks which are not stable, we go back to the "Authentication mark embedding" routine, and perform the embedding again.

When the algorithm loops, by de-quantizing $c(i,0,d_l)$ not with n, which is the quantized value of Qe_l, but with a numeric number whose absolute value is smaller than n, $|c(i,0,d_l)|$ will become close to 0 and the probability of overflow or underflow can be reduced when BMP transformed again. But if we set n too small, the fidelity will be degraded, so, we should gradually reduce n as it loops. Also, for k such as $k \notin \{d_l\}$, we can make them stable quickly and without degrading the fidelity by gradually making $|c(i,j,k)|$ approach 0 as it loops.

When all the blocks become stable after several loops, the embedding procedure for the image I is completed.

Embedding the hash value: Using all the quantized DCT coefficients which belong to d_l except for the $\boldsymbol{A}$ area, we calculate the hash value E as

$$E = H(K_e, \bigcup_{i \notin \boldsymbol{A}, l} c_q(i,l)) \tag{7}$$

where

$$c_q(i,l) = [(c(i,0,d_l) + \alpha)/Qe_l] \tag{8}$$

and write E to the $\boldsymbol{A}$ area. Here, α is the offset for the quantization, and in this case, $\alpha = 0$.

One method to embed E into $\boldsymbol{A}$ area is to modify the LSB (Least Significant Bit) of the DCT coefficients, which is called the LSB method, and in this case, all the blocks in the $\boldsymbol{A}$ area must be stable after the hash value is embedded. One method to embed the hash value in this way is to use the LSB method by looping between the procedure which corresponds to the section between “Authentication mark embedding” and “Post-embedding process” in Fig 1.

3.2 Detection

Detection consists of two major phases, ”Pre-detection processing” (decoding, BMP$\rightarrow$JPG’ transforming, inverse-calculation of the embedding Q step) and ”Alteration detection” (Fig 2). We explain the detection procedure using Fig 2. We skip BMP$\rightarrow$JPG’ transformation and JPEG’ decoding because these were already discussed in the Embedding subsection.

Inverse calculation of Q_e: We need to know Q_e in order to correctly detect the alteration correctly of the target image I', but when the format of I' is BMP, no Q table is included, and even for JPG, the attached Q table does not always reveal the Q_e values. But, in our method, the Q_e’s are large enough to satisfy Eq. (2), so, we can recover the Q_e values from the histogram of the DCT coefficients of $\{d_l\}$. For example, in the bottom graph of Fig. 6, $Qe_l = 12$.

Alteration detection: We compare the hash value E', which is calculated from the coefficients of $\{d_l\}$s in the area which don’t belong to $\boldsymbol{A}$ in I', with E, which is embedded in $\boldsymbol{A}$, and determine whether or not some intentional alterations have been performed to I'. When we calculate E' in Eq. (7), we set α as

$$\alpha = \begin{cases} Qe_l/2, & c(i,0,d_l) > 0 \\ -Qe_l/2, & \text{otherwise}. \end{cases} \qquad (9)$$

4 Experiment

We took a picture of a car using a digital camera (JPEG, compression ratio = 1/20, Fig. 3, left), and created an embedded JPEG file following the procedure discussed in Section **3.1** (Fig 3, right), performed JPG to RGB transformation, and compared the distribution histogram of the R,G,B values of the DCT coefficients. In our experiments, we didn’t actually embed the hash value, because our purpose at this time was to verify the characteristics of the images before and after embedding the authentication mark, which corresponds to just before “Hash value embedding” of Fig. 1. We used $\{d_l\}$={9,10,17} (Fig. 4) and $\delta = 7$.

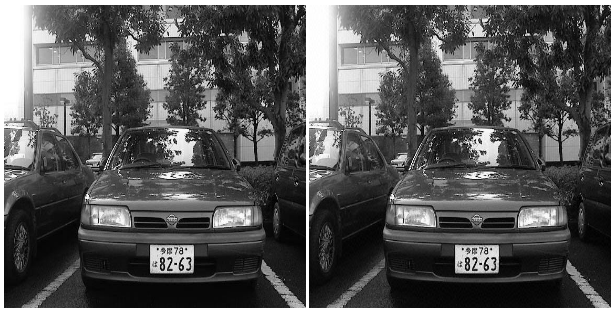

Figure 3. left: original image, right: embedded image

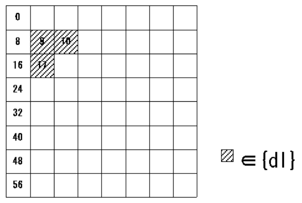

Figure 4. DCT block

Fig. 5 shows the distribution histograms of the coefficients of the R factor without performing overflow or underflow processing after the JPG to RGB transforming before (top) and after (bottom) embedding, and Table 1 shows the ratio of overflows (more than 255) or underflows (less than -1) of the R,G,B values before and after embedding. From these figures and tables, we find that many overflows or underflows are observed only by using the iDCT, YUV to RGB transformation, even for JPEG images into which the authentication marks have not been embedded. A few overflows or underflows are observed even after embedding, but they are few enough not to violate the stable condition. Fig. 6 shows the distribution histograms of the coefficients of DCT index $d_l = 9$ before (top) and after (bottom) JPG to RGB + RGB to JPG’ transformations, and Table 2 shows the ratios of the coefficients which don’t satisfy Eq. (6) at $\epsilon = 3$, $d_l = 9, 10, 17$, before embedding. The image used

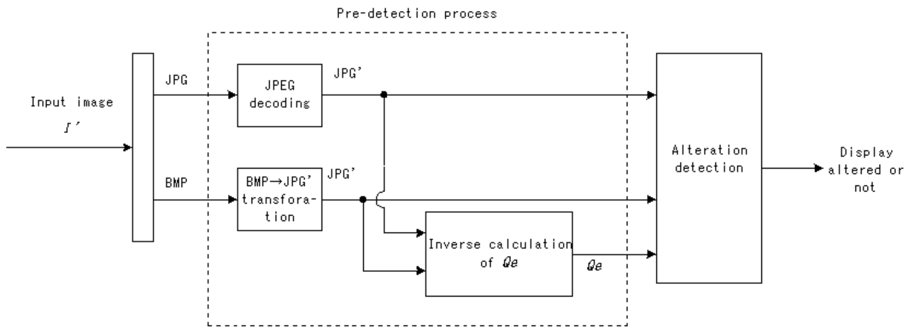

Figure 2. Detection procedure

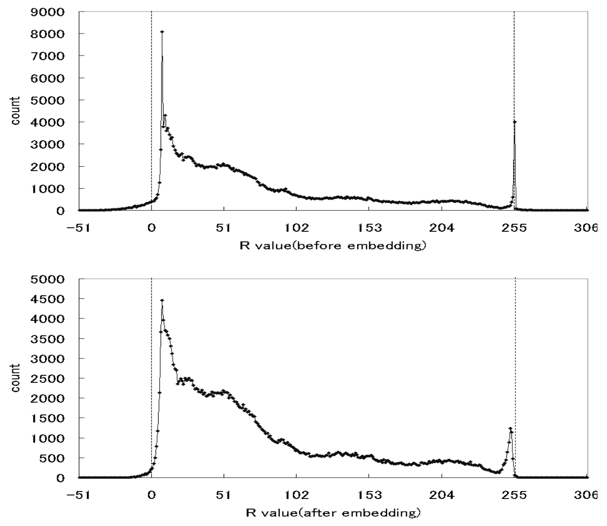

Figure 5. R factor coefficients distribution histogram (Upper: Before embedding, Lower: After embedding, dotted vertical line: overflow or underflow threshold)

had the value $Q_l = 6$ for $d_l = 9$, so $Qe_l = 12$ from Eq. (2). As you can see from these figures and tables, many coefficients have different Q values from simply performing iDCT + YUV to RGB transformation to the image before embedding. Also we can see that after embedding, all the coefficients which belong to $\{d_l\}$ satisfy Eq. (6), that is, they are close to nQe_l.

The average embedding and detection times for one JPEG image were 3.0 sec and 0.2 sec, respectively, using an IBM Intellistation (Pentium II 300 MHz), running under Windows 98, using 50 different 1024×768 pixel images.

		R	G	B
Before embedding	overflow	0.31	0.50	1.09
	underflow	1.75	0.71	0.45
After embedding	overflow	0.05	0.10	0.36
	underflow	0.37	0.11	0.12

Table 1. The ratio of overflow or underflow of the coefficients before and after embedding (percent)

d_l	9	10	17
ratio	1.90	9.44	1.40

Table 2. The ratio of coefficients which are not close to Q value, before and after embedding (percent)

5 Discussion

In order for intentional alterations to be detected, the value $c_q(i,l)$ must be changed for at least one (i,l) pair. Now we define $p_{l,n}$, P_t to be the probability that $c_q(i,l)$=n holds, and the probability of detecting the alteration when the alteration involves b blocks, respectively. Then, assuming that the histograms of the quantized coefficients of $\{d_l\}$ before and after the alteration are not changed, the following equation holds:

$$P_t = 1 - (\prod_l (\sum_n p_{l,n}^2))^b \tag{10}$$

By using the observed $p_{n,l}$ from the embedded image used in Section **4**, $P_t = 1 - 2.78 \times 10^{-20}$ at $b = 11$,

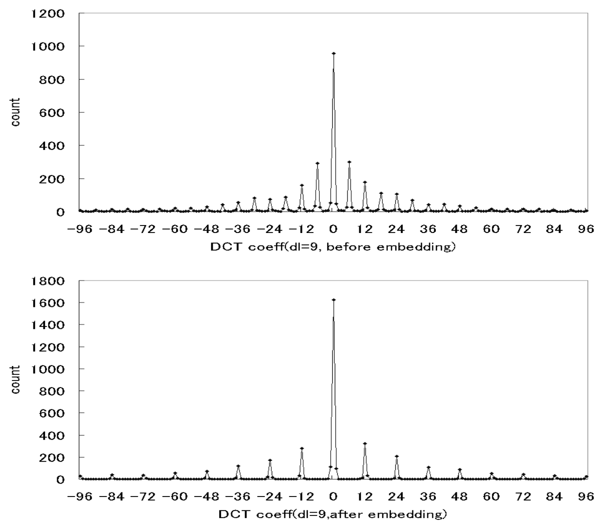

Figure 6. DCT coefficients distribution histogram (d_l=9, Upper: before embedding,Lower: after embedding)

which is close to $1-2^{-64}(=1-5.24\times10^{-20})$, which is the alteration detection ratio when we use a hash with a 64 bit key. As an example, $b\simeq 40$ for the case of altering the number plate of the car of Fig. 3, so, for this example, P_t would be much smaller. In order to make this system even more secure, we need some changes such as increasing the number of $\{d_l\}$ in case the attackers guess $\{d_l\}$ (very little impact on the fidelity even if all the AC indexes are used), or changing $\{d_l\}$ per block using some pseudo random number, or so forth.

6 Application

In this paper, we focused only on the detection of the altering, not on the detection of the location of the alteration, but we can create images which are robust to format transformations but can also detect the location of the alteration, while still holding the FP ratio constant, for example, by embedding authentication marks for each block with the method we have introduced, so, each coefficient is close to Qe_l, and by adding some rules for LSB or large and small relationships of the DCT coefficients [3].

7 Conclusion

In this paper, we have introduced an alteration detection method using the characteristics of the Q values in the compressed domain, and showed the experimental results and timings. One of the prominent characteristics of our method is that the authentication mark is robust to format transformations which are accompanied by overflow or underflow, but is very sensitive to even a small deliberate alteration regardless of the image format. Because most deliberate alteration is actually performed to an uncompressed format such as BMP, it's a useful feature that images whose authentication marks are embedded in the JPEG domain can reveal the alterations from the BMP domain.

References

[1] K. Toyokawa, N. Morimoto, S. Tonegawa, K. Kamijo, and A. Koide, “Secure digital photograph handling with watermarking technique in insurance claim process,” in *Proceedings of SPIE*, pp. 438-445, San Jose, CA, 2000.

[2] S. Ikeno, et al., “Cryptographic Theory”, the Institute of Electronics, Information and Communication Engineers

[3] S. Tonegawa, K. Kamijo, T. Nakamura, N, Morimoto, and A. Koide, “Alteration Location Detection Method using DataHiding Technology”, in *Proc. of IPSJ 61st Annual Conf*, Vol. 3, pp. 35-36, 2000.

[4] K. Kamijo, S. Tonegawa, K. Toyokawa, N. Morimoto, and A. Koide, “A Method to Protect and Detect Alterations of Digital Photographs in Insurance Claim Process (II),” in *Technical report of IEICE, IN99-97,* Vol. 99, No. 591, pp. 55-60, 2000.

[5] K. Kamijo, “A DataHiding Method Robust to Format Transformation”, in *Proc. of IPSJ 61st Annual Conf*, Vol. 3, pp. 37-38, 2000.

[6] ISO/IEC 10918-1,“Information technology -Digital Compression and Coding of Continuous-tone Still Images”