Journal of Applied Sciences

ISSN 1812-5654

Journal of Applied Sciences 13 (22): 5260-5264, 2013
ISSN 1812-5654 / DOI: 10.3923/jas.2013.5260.5264
© 2013 Asian Network for Scientific Information

Watermarking Image-based Vehicle Parking Enforcement Program for Preventing Evidence Manipulation

[1]Geonam Kim, [1]Hong-Mok Choi, [2]Doyeon Kim and [1]Jaejoon Kim
[1]School of Computer and Communication Engineering, Daegu University, Gyeongsan, 712-714, Korea
[2]School of Computer and Information Technology, Daegu University, Gyeongsan, 712-714, Korea

Abstract: In an age where automobiles have become the most used method of transportation, a problem of illegal parking has arisen with insufficient police officers to enforce the law. Commercial or government-based parking enforcement programs were developed in order to lessen the load of the police deployed. The main objective of this study is to prevent false evidence from entering the system. Digital watermarking technology was imported via LSB transformation under law enforcement. Among various approaches, the Least Significant Bits (LSB) transformation scheme is easy and simple. An Android smartphone with an app installed was used with a PC desktop that is able to check the key stored from the application within a Wi-Fi zone. An experiment was performed to extract the key from an image taken from the smartphone to prevent false evidence.

Key words: Mobile parking enforcement system, evidence manipulation, watermarking technique, encryption key, LSB transformation

INTRODUCTION

In an age where automobiles have become the most used method of transportation, the problem of illegal parking has arisen with insufficient police officers to enforce the law. Even if the police are called, there is too much time wasted in just deploying officers to the scene. A parking enforcement program was developed in order to lessen the load of the police deployed (Kim and Kim, 2013). In the program, a photo of an illegally parked car is first taken and sent to the police. If the car remains after the time allowed, then a second picture is taken and sent to the police. The information is sent to a police server where the information is processed and used by the police during investigations. However, it is difficult for the police to determine how much of the information sent to them should be trusted.

Since 1993, when IBM and Bell-South Cellular Corp. developed the first smartphone 'Simon' (O'Malley, 1994), there has been 20 years of innovation and development of applications. These applications now allow people with even basic knowledge to manipulate images and submit false evidence. In order to prevent false evidence, we decided to import digital watermarking technology. Since the late 1990s, there has been much advancement in the field and the technology is now being used extensively. This study validates an implementation of a digital watermarking method applied at the space level to prevent false evidence.

The study is organized as follows. In the next section, fundamental digital watermarking technologies are reviewed. In Section 3, the insertion and detection of an encryption key in an image are presented. Section 4 presents the experimental results and finally, the study is concluded in section 5.

REVIEW OF WATERMARKING TECHNOLOGY

Since the late 1990s, watermarking has been extensive researched and is now widely used. Watermarking generally can be divided into two categories according to whether it is performed at the space level or the frequency level. For watermarking at the space level, least significant bits (LSB) watermarking is extensively used and for the frequency level, Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT) watermarking are widely used.

LSB transformation: For watermarking technology at the space level, the LSB method is the most widely used (Cox *et al*., 2002). It works by transforming the LSB and putting it into the image. The watermark is finished by dividing the information to be added into single bits then putting them into the LSB of the original image (Celik *et al*., 2002; Chang *et al*., 2000). However, while the process is quick and easy, it has a weakness in that it can be decoded easily. It is easy to obtain the watermark payload using a backtracking algorithm.

Corresponding Author: Jaejoon Kim, School of Computer and Communication Engineering, College of Information and Communication Engineering, Daegu University, Gyeongsan, 712-714, Korea

DCT transformation: The DCT (Discrete Cosine Transform) method is widely used at the frequency level. Chu (2003) found that the DC component of the DCT gives a better sense of space compared to the AC component. He divided the watermark payload into a rough part and smooth part and then inserted all the watermark signals into the DC area of the DCT. Huang *et al*. (2000) experimented with acquiring each individual DCT block and with the insertion of the coefficients of each DCT block as a watermark. Piva *et al*. (1997) inserted the coefficients of the middle frequency from a predetermined zigzag scanning process. Barni *et al*. (1998) divided an image into each frequency and selected the middle frequency to insert the watermark. While inserting the watermark, they used a method to spread the watermark over a larger region to reduce visibility. The best part of this method is that the original image is not required when extracting the watermark.

DWT transformation: The DWT (Discrete Wavelet Transform) algorithm is one of the most widely used algorithms in the frequency space for watermarking technology. The basic idea of the algorithm is that a video or image is divided into different sub-bands, such as LL, LH, HL, or HH and then put back into the sub-band after the watermarking payload is transformed. Barni *et al*. (2001) first introduced the method and used it to insert subtle coefficients of the sub-band into the watermark. In order to prevent the length of the watermark from being detected, the mask is adjusted. Lumini and Maio (2000) divided a video by using a vertical and horizontal sub-sampling filter bank into the individual sub-bands and then dividing the LL part once more by using a level 2 wavelet, followed by inserting the watermark payload.

ENCRYPTION KEY INSERTION AND DETECTION

The present approach uses the LSB transformation method to prevent the manipulation of image evidence. Although the LSB transformation method is invisible, it is not tenacious and it creates a watermark that can be easily broken. However, since the manipulation of image evidence is being considered, the fact that the watermark can be easily deciphered is not a problem. The key is arranged in an array and it could be detected even without the original image. Figure 1 shows a diagram of inserting the watermark and detecting the key.

Inserting encryption key: The key is created from 8 random letters obtained by the ASCII value in between A-Z and a-z. One letter can be divided into 8 pieces with 1 bit each and put into the binary system. This binary code is inserted into each pixel of the original image and

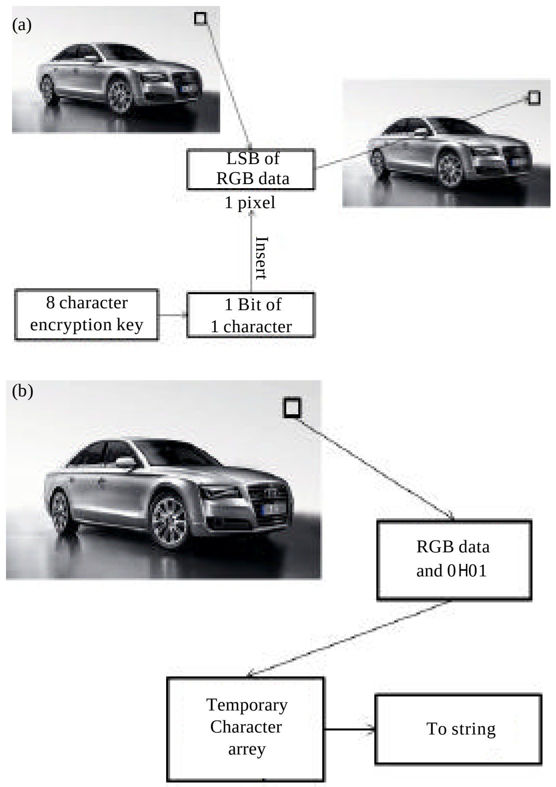


Fig. 1(a-b): A diagram of (a) Inserting watermark and (b) Detecting the key

the pixel color information is inserted in the LSB. First, we bring the ARGB (Alpha Red Green Blue) pixel information and then we use the LSB to clear as 0. Then, each letter is transformed into a binary code so that the variable can be temporarily stored. Finally, using the ARGB information and OR calculation, 1 bit of a letter is hidden and then stored back in the image. Figure 2 illustrates the process of inserting the LSB.

The first location is considered as the coordinate (0, 0) for the pixel and the y-axis will contain a total of 8 squares with 1 bit of information for each letter. When inserting the code key into the image, the size of the image in the x-axis must be larger than the size of the string that will be inserted. The y-axis must be larger than the 8 pixels. Figure 3 shows an illustration of this process.

In Fig. 3, the ‘encryption Key’ indicates the key before it has been inserted into the image and ‘Origin_img’ is the information of the original image. A new image is not made for the LSB transformation; instead, the LSB-transformed ARGB information is inserted directly into the original image.

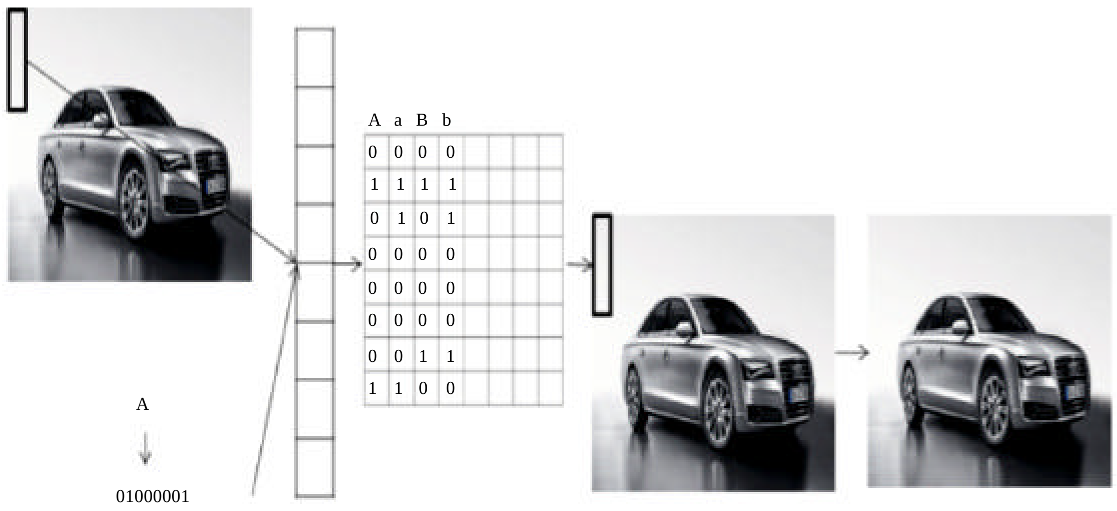


Fig. 2: A process of encryption key insertion

```
public Bitmap insertEncryptionKey{} {
        int x = 0, y = 0;
        for (int j = 0; j< encryptionKey.length{}; j++) {
                for (int k = 7; k >= 0; k--) {
                        int ARGB = Origin_img.getPixel(x, y);
                        ARGB = ARGB & 0x FFFFFFFF;
                        int temp2 = encryptionKey.charAt(j) >>> k;
                        int temp2 = temp2 & 0x01;
                        ARGB = ARGB l temp2;
                        Origin_img.setPixel(x, y, ARGB);
                        y++;
                }
                 y = 0;
                 x++;
        }
         return img;
}
```

Fig. 3: A code for watermark insertion

Detecting encryption key: The basics of extraction work in the opposite direction of when a watermark is inserted into the image. First, the image is loaded and its dimensions are determined. Since the key was inserted starting from (0, 0), extraction is also started from (0, 0). After obtaining the pixel color information, all the bits except the LSB are cleared to ‘0’. While incrementing the pixel position by one on the y-axis, the integer obtained through LEFT_SHIFT and OR calculation is temporarily stored. The 8-bit binary code is then changed into letters and stored in a temporary string. This process is repeated as many times as the length of the string and store the information one by one in a matrix that is later changed into a string. Figure 4 shows the coding for this process.

```
public String Decording() {
        int  = × 0, int y = 0;
        int width = bufImg.getWidth(null);
        int hight = bufImg.getHight(null);
         char [] t_char = new char[10];
         int [] argb = new int[width*height];
        bufImg.getRGB(0, 0, width, height, argb, 0, width);
        while( ×  < 8 ) {
                for (int j = 7; j >= 0; j--) {
                         int ARGB = argb[×+ (y * width)];
                        tempARGB = tempARGB & 0×00000001;
                        int temp = ARGB << i;
                        t_int = t_int l temp;
                         y++;
                }
                t_char[×] = (char) t_int;
                t_int = 0×00;
                 y = 0;
                x++;
        }

        return enaryptionKey;
}
```

Fig. 4: Decoding process for detecting an encryptionKey

EXPERIMENTAL RESULT

As shown in Fig. 5, an Android smartphone with the app installed was used with a PC desktop that is able to check the key stored from the application. In order to ensure safe data transfer, experiments were conducted within a Wi-Fi zone. An experiment was performed to extract the key from the image that was taken from the smartphone.

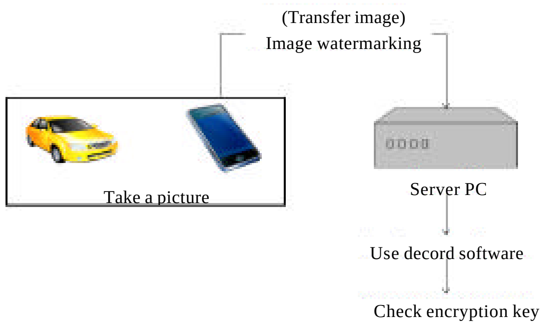
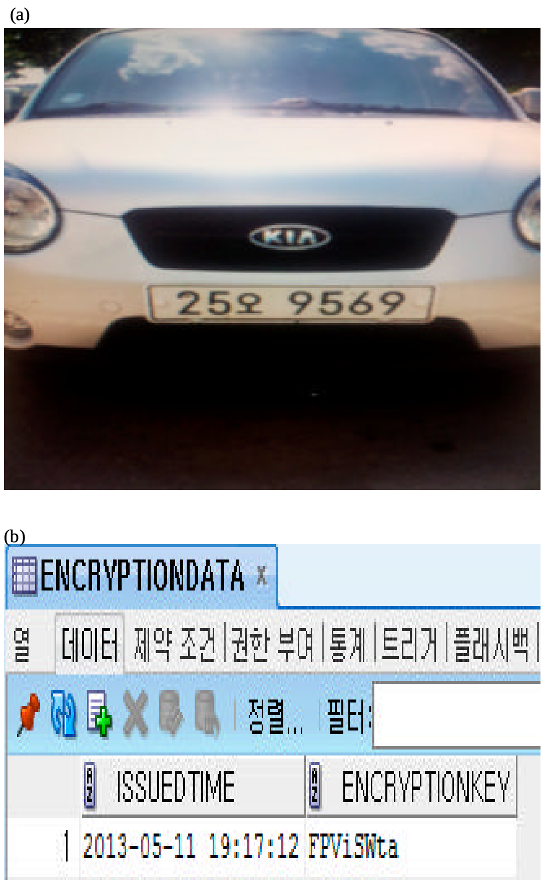


Fig. 5: Experiment procedures



	ISSUEDTIME	ENCRYPTIONKEY
1	2013-05-11 19:17:12	FPViSWta

Fig. 6(a-b): An experimental process, (a) An inserted image with encryption key and (b) Encryption key stored in DB

Parking enforcement program: When running the app and taking a picture, the smartphone does not make a key and insert it into the image immediately. Figure 6a shows that it is difficult to differentiate whether the image has the key or not. Also, even if modification or tampering with the image is attempted, since the image is not formed in the smartphone, it is difficult to manipulate the image.

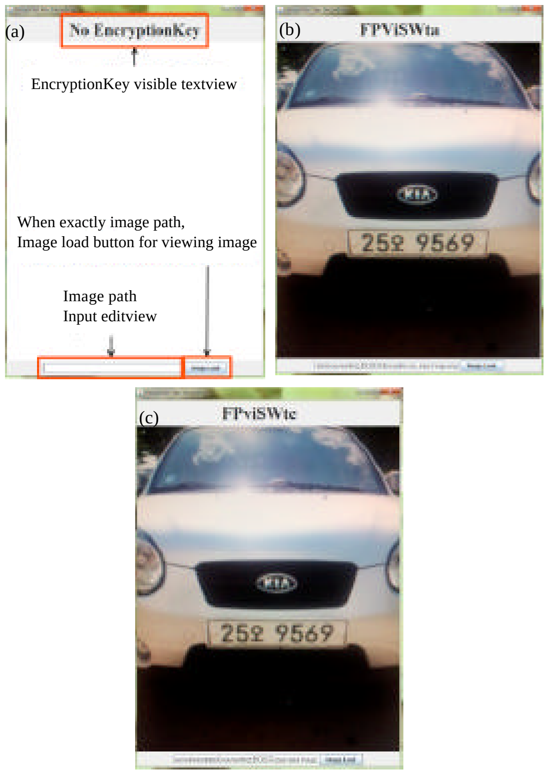


Fig. 7(a-c): Results for insertion and detection of encryption key, (a) Graphic user interface, (b) An image without and (c) An image with encryption key

Figure 6b shows the key that is formed right after the image is taken and stored in the Database. The key is formed in order to check the precision of the key obtained during extraction and stored in the database. For this experiment, the key 'FPViSWta' is used.

Checking encryption key: Figure 7a shows the simple graphical user interface made to check whether the key is formed. The top is labeled so that the key can be checked. A space was made in the middle to form the image and the bottom has an input to load the image.

When there is no image loaded, as in Fig. 7a, the label reads 'No EncryptionKey'. Figure 7b shows the result of the app running from the sample image. After loading the image, the key that was inserted into the image shows up in the label. As expected, it can be seen that the key is 'FPViSWta'. The key was inserted properly into the image and during extraction, the intact key could be extracted.

CONCLUSION

A key was inserted to prevent the manipulation of evidence and there were no problems in extracting the key. However, one possible problem is that the position at which the key is inserted is fixed, so if the image is manipulated in other places apart from the key region, the keys of the original image and the manipulated image may end up being the same. In order to fix this problem in future works, the key can be spread out so as to cover a broader region of the image, or the key can be inserted into specific areas of the original image to suit the purpose. Thus, if there is an improvement in the placement of the key to suit various purposes that require watermarking, this technology can be used in the field in the near future. Also, in this study a key was produced from a combination of upper and lower case letters. For future works, if the key inserted could work as a coded message, it can be used to prevent manipulation of the image more effectively.

ACKNOWLEDGMENT

This study was supported by the Research Program of Daegu University in 2012.

REFERENCES

Barni, M., F. Bartolini and A. Piva, 2001. Improved wavelet-based watermarking through pixel-wise masking. IEEE Trans. Image Process., 10: 783-791.

Barni, M., F. Bartolini, V. Cappellini and A. Piva, 1998. A DCT-domain system for robust image watermarking. Signal Process., 66: 357-372.

Celik, M.U., G. Sharma, E. Saber and A.M. Tekalp, 2002. Hierarchical watermarking for secure image authentication with localization. IEEE Trans. Image Process., 11: 585-595.

Chang, C.C., K.F. Hwang and M.S. Hwang, 2000. A digital watermarking scheme using human visual effects. Imformatica, 24: 505-511.

Chu, W.C., 2003. DCT-based image watermarking using subsampling. IEEE Trans. Multimedia, 5: 34-38.

Cox, I.J., M.L. Miller and J.A. Bloom, 2002. Digital Watermarking. Morgan Kaufmann Publishers, San Francisco, USA.

Huang, J., Y.Q. Shi and Y. Shi, 2000. Embedding image watermarks in dc components. IEEE Trans. Circuit Syst. Video Technol., 10: 974-979.

Kim, D. and J. Kim, 2013. Implementation of efficient parking enforcement system using smartphone. Int. J. Contents, 9: 26-32.

Lumini, A. and D. Maio, 2000. A wavelet-based image watermarking scheme. Proceedings of the International Conference on Information Technology: Coding and Computing, March 27-29, 2000, Las Vegas, NV., USA., pp: 122-127.

O'Malley, C., 1994. Simonizing the PDA. Byte, 12: 145-148.

Piva, A., M. Barni, F. Bartolini and V. Cappellini, 1997. DCT-based watermark recovering without resorting to the uncorrupted original image. Proceedings of the International Conference on Image Processing, Volume 1, October 26-29, 1997, Santa Barbara, CA. USA., pp: 520-523.