Chapter 3
Copyright Protection of MPEG-4 Videos by Active Watermarking and Limited Play Counts

3.1 Introduction

Due to the growth of computer networks and audio/video compression technology, more and more digital videos are shared on the Internet in various forms. But the owners of videos usually want to keep the ownership and copyright of their videos protected. In order to solve this problem, an active visible watermarking technique is proposed for copyright protection of MPEG-4 videos in this study and is described in this chapter.

Roughly, we can watch videos on the Internet easily through several ways, such as streaming, downloading, etc. Streaming means that users continuously receive the data transferred in a stream of packets that are interpreted and rendered by a software application as the packets arrive. Although the streaming technique seems to be a trend nowadays, there is an implicit defect that the presentation of videos will not be smooth if the data receiving speed is slower than the software processing speed due to the limitation of the network bandwidth.

Lo and Tsai [14] proposed an active watermarking technique for online copyright protection of MPEG videos using an ActiveX control program embedded in the web page as an *active agent*. Users watch videos on web browsers supporting the ActiveX technology. As mentioned before, the smoothness of watching online videos depends

on the network bandwidth. In order to improve this defect, we propose an idea in this study, that is, users can download videos to their computers and watch them without consideration for network problems. Moreover, we can even impose the limit of the play count of a video on a user and claim the ownership of the video by the supplier immediately with visible watermarks as the play count for the user runs out.

The remainder of this chapter is organized as follows. In Section 3.1.1, some related problem definitions are given. In Section 3.1.2, the basic idea of the proposed scheme and the system configuration are presented. In Section 3.2, the scheme for video play-count limitations is described. In Section 3.3, a visible watermark embedding method is proposed, and a corresponding visible watermark removal method is stated in Section 3.4. In Section 3.5, several experimental results of the proposed method will be shown. Finally, some discussions and a summary will be made in the last section of this chapter.

3.1.1 Problem Definition

A video downloaded from a website can be duplicated into several copies at a local computer or sent to others who also want to watch the video. Therefore, the first problem is how to prevent a video from being copied and how to implement a scheme for limiting the play count. The second is how to embed a visible watermark into an MPEG-4 video with a secret key. The final problem is how to remove the embedded visible watermark to recover the original video actively without manual manipulation.

3.1.2 Proposed Idea and System Configuration

First, a user has to register the membership at a website, and receives a distinct username and a password. After that, the user is allowed to browse movies listed on the web page and download favorite ones to his/her local computer after paying a reasonable price. We will randomly generate a secret key for each user and use this

secret key to embed a visible watermark into the video.

While a user wants to watch a downloaded video, he/she has to download an active player proposed in this study for video playback instead of the Windows media player or other video players. Otherwise, the embedded visible watermark cannot be removed for copyright protection of the downloaded video. The main task of this player designed in this study is to check and update the play count of the corresponding video automatically and remove the visible watermark by checking invisible signals embedded in the watermarked video. The system configuration is shown in Figure 3.1.

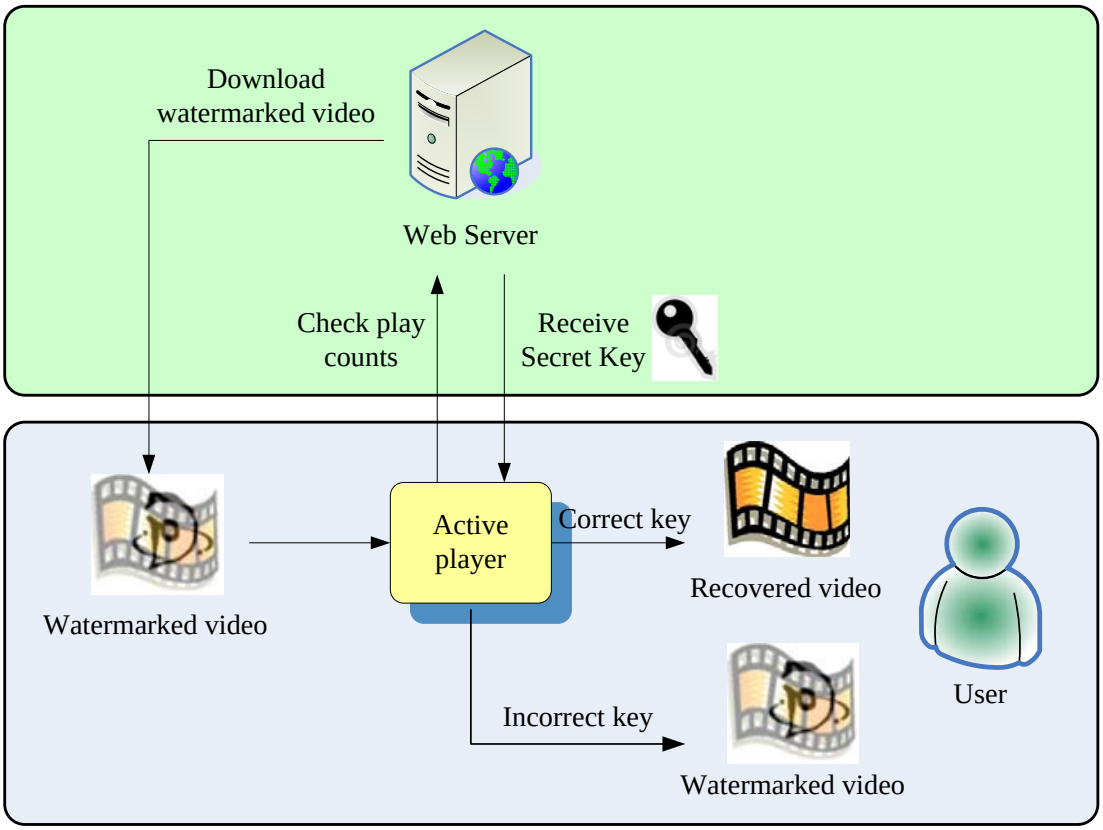


Figure 3.1 System Configuration

3.2 Scheme for Video Play-Count Limitation

In Section 3.2.1, the basic idea of the proposed scheme for video play-count limitation will be described. An illustration of the checking process of the video play count is shown in Figure 3.3. In Section 3.2.2, a detailed algorithm will be described.

3.2.1 Idea of Proposed Scheme

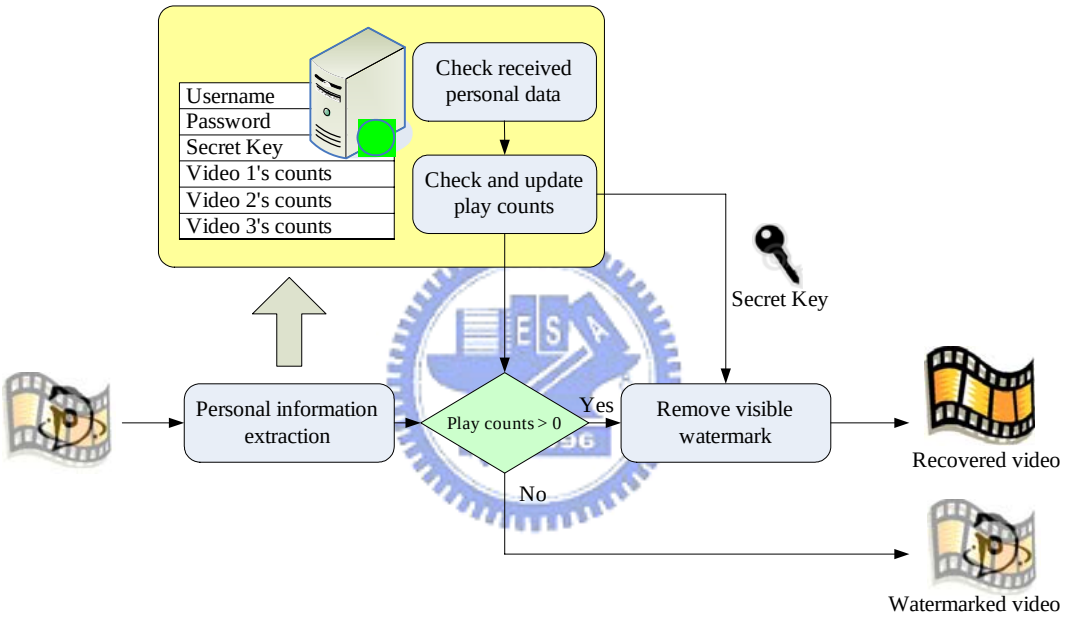


Figure 3.2 Checking Process of video play count.

For the sake of protecting the copyright of a video and avoiding a video being duplicated into several copies, we propose an idea to take control of a downloaded video via a remote server. While a user requests the download service from the video supplier, a play count with a fixed value is assigned to each requested video by the server. Then we embed the related personal data, including the username and the video serial number, into the requested video so that we can easily update the play count of the corresponding video.

Each time the user plays the video with the active video player, the player will

actively connect to the server and check whether the play count for the user is greater than zero or not. If so, the server will decrement the play count by one and return a secret key to the active player for removal of the visible watermark. Otherwise, the video will be covered with a visible watermark by the supplier for the purpose of claiming the ownership of the video and luring the user to pay for more play counts. Therefore, supposing that the user wants to duplicate their videos into many copies, we can prevent this forbidden attempt by an online checking process performed on the server. A flowchart of the online checking process is shown in Figure 3.2.

3.2.2 Detailed Algorithm

In order to retrieve the data quickly, we choose the first I frame to embed the personal data. As mentioned in Chapter 2, the compression technique of the I frame in the MPEG-4 standard is similar to previous standards like MPEG-1 and -2. The DCT-based method is adopted in this study to hide data into the I frames. Each 8×8 luminance block which can be used to hide at most four bits of data by making a slight modification on a selected pair of DCT coefficients is illustrated in Figure 3.3. Let (x, y) denote the location of a coefficient. In the proposed process of embedding personal data, we choose a fixed pair of DCT coefficients to embed one bit of hidden data. In addition, we do not need any secret key in this process since the username and the video serial number are public between the client and the server, and the result of this process will not influence the result of embedding or extracting visible watermarks. A flowchart of the hiding process for each intra-coded macroblocks in the first I frame is shown in Figure 3.4. A detailed algorithm of the process is described in the following.

(*x*, *y*)	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
2	16	17	18	19	20	21	22	23
3	24	25	26	27	28	29	30	31
4	32	33	34	35	36	37	38	39
5	40	41	42	43	44	45	46	47
6	48	49	50	51	52	53	54	55
7	56	57	58	59	60	61	62	63

Figure 3.3 An illustration of selected pair of DCT coefficients in an 8×8 luminance block.

Algorithm 3.1: Embedding the personal data into the first I frame.

Input: the first I frame *F*, a username *N*, and a video serial number *S*.

Output: an I frame *F*' with the personal data embedded.

Steps:

1. Transfer *N* into the binary form N_b as $N_b = n_1 n_2 n_3 \ldots n_{L_1}$, where L_1 is the length of N_b
2. Transfer *S* into the binary form S_b as $S_b = s_1 s_2 s_3 \ldots s_{L_2}$, where L_2 is the length of S_b.
3. Calculate the sum of L_1 and L_2, and transfer it into the binary form L_{sum}.
4. Combine L_{sum}, N_b and S_b sequentially to form a new binary string *B*.
5. For each 8×8 block used to hide the personal data, select two DCT coefficients located at (6, 4) and (5, 5) in the 8×8 DCT matrix as a pair (C_1, C_2) to embed a bit b_i of *B* according to the following rule.

   (1) When b_i = 0:

$$\begin{cases} if \;\; C_1 < C_2, \; then \;\; swap \;\; C_1 \;\; and \;\; C_2; \\ if \;\; C_1 = C_2, \; then \;\; set \;\; C_1 = C_2 + T_1. \end{cases} \tag{3.1}$$

(2)When b_i = 1:

$$\begin{cases} if \;\; C_1 > C_2, \; then \;\; swap \;\; C_1 \;\; and \;\; C_2; \\ if \;\; C_1 = C_2, \; then \;\; set \;\; C_2 = C_1 + T_1. \end{cases} \tag{3.2}$$

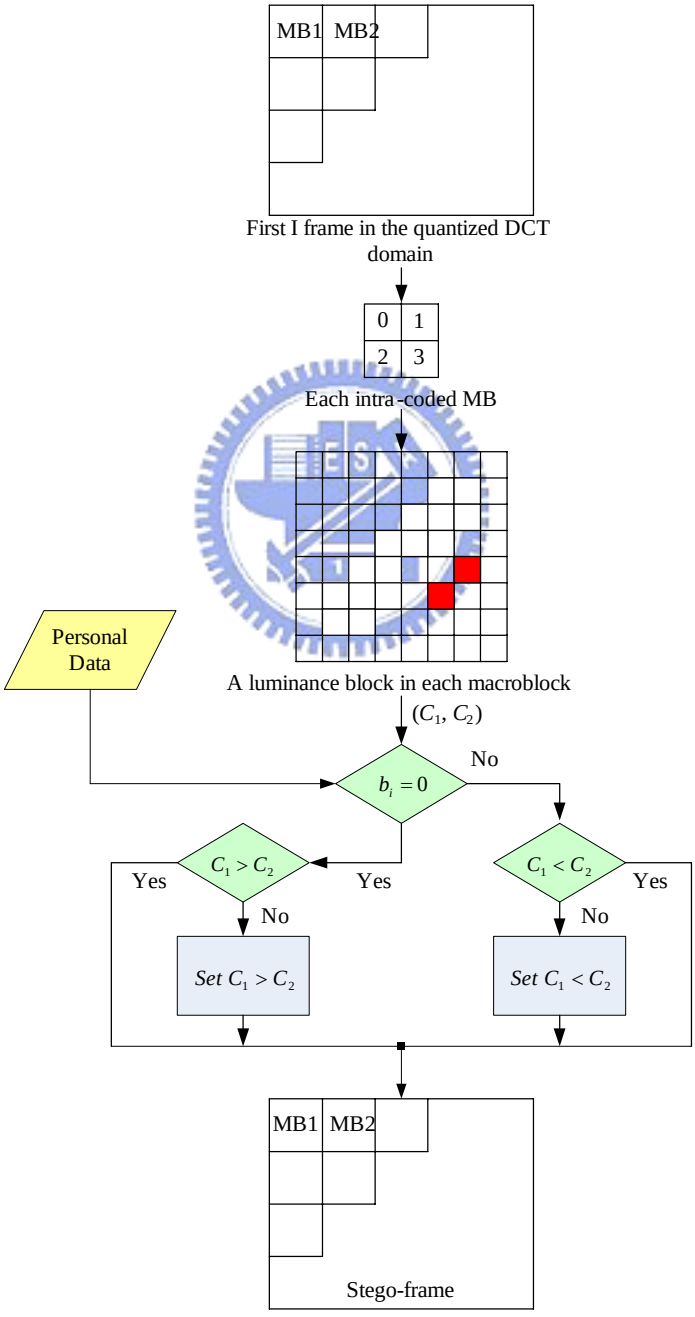


Figure 3.4 Flowchart of process for hiding personal data.

3.3 Embedding Visible Watermarks in MPEG-4 Videos

In Section 3.3.1, the proposed idea of embedding visible watermarks is stated. In Section 3.3.2 and 3.3.3, the process for embedding watermarks in I, P, and B frames is described.

3.3.1 Proposed Idea

A watermark to be embedded into a video is assumed to be a binary image (with only black and white pixels). Each pixel in the watermark will be considered as a black watermark pixel or a white one according to their binary values 1 or 0. The embedding process for I, P, and B frames in a given video utilizes each 8×8 luminance block of the video to embed a watermark pixel. Besides, we also embed certain invisible signals generated by the secret key into each 8×8 block to determine whether this block is watermarked or not. And the invisible signals are embedded by making a slight modification of certain DCT coefficients.

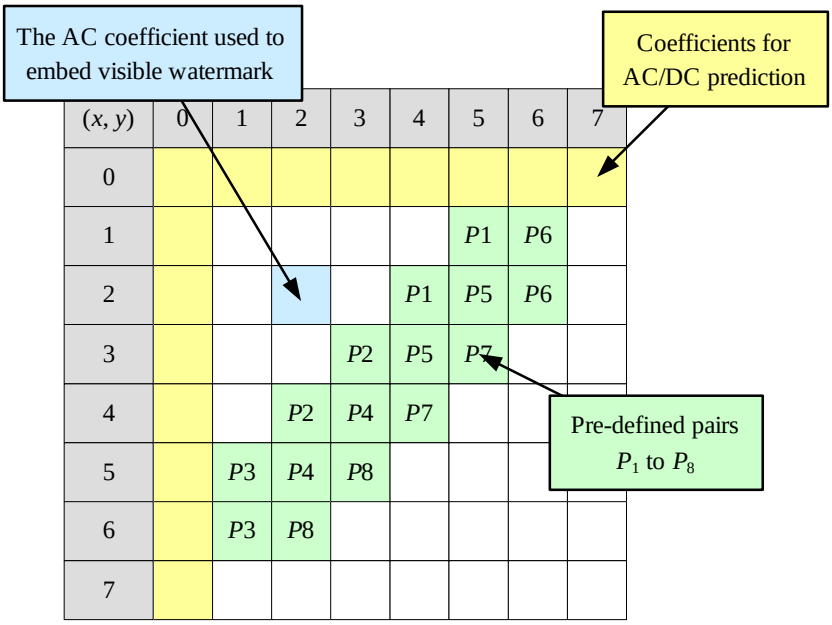


Figure 3.5 Location of significant coefficients in the 8×8 luminance block.

Generally speaking, the DCT coefficients in the middle frequency band are usually modified for data hiding because slight changes of such coefficients are imperceptive to human eyes. Therefore, as illustrated in Figure 3.5, we select eight pairs of coefficients in the middle frequency band according to the zig-zag scanning order, and each pair is randomly selected for hiding the invisible signals.

3.3.2 Process for Embedding Visible Watermarks in I Frames

A new technique, called AC/DC prediction, is adopted in the MPEG-4 standard and results in higher compression efficiency than previous standards. The AC/DC prediction is used for I frames and performs prediction on the first row and the first column of the DCT coefficients in each 8×8 block according to the transformed blocks situated on the left, top-left and top of the current block. It implies that the DC coefficient of each 8×8 block in each I frame cannot be modified for embedding a watermark pixel [5] because it will cause a *drift problem* which means a prediction error. Try to imagine that an 8×8 block which is not watermarked in a watermarked image frame is taken as input to the MPEG-4 decoding process. When the decoder performs the AC/DC prediction, if this block is decoded with a reference to a watermarked block, then this block will also be watermarked. Hence, we propose an idea that we not only create a special visible watermark but also prevent this kind of drift problem. We choose the AC coefficient located at (2, 2) in the 8×8 matrix shown in Figure 3.5 to embed the watermark pixel, and it results in a *twinkling effect* to prevent users from enjoying the video. A flowchart of the embedding process for I frames is shown in Figure 3.6 and a corresponding detailed algorithm of the process is described in the following.

Algorithm 3.2: Watermark embedding process for I frames.

Input: an I frame F in the quantized DCT-domain, a secret key R, and a binary watermark image W.

Output: a watermarked frame F' with a visible watermark W' similar to W.

Steps:

1. Map each watermark pixel w_{ij} to the corresponding 8×8 block B_{ij} of F.
2. For each 8×8 luminance block B_{ij} of F, combine the secret key R and the position P of B_{ij} to form a seed for a random number generation. Select one pair of coefficients (C_1, C_2) from eight pre-defined pairs of coefficients according to the result of random number generation.
3. Embed the invisible signals, each of which represents whether the watermark pixel w_{ij} of W to be embedded into B_{ij} is black or white, by changing the relation between C_1 and C_2 according to the following rule.

   (1) When w_{ij} is black:

$$\begin{cases} if\ \ C_1 < C_2,\ then\ \ swap\ \ C_1\ \ and\ \ C_2; \\ if\ \ C_1 = C_2,\ then\ \ set\ \ C_1 = C_2\ +\ T_2, \end{cases} \tag{3.3}$$

   where T_2 is a pre-defined threshold.

   (2) When w_{ij} is white:

$$\begin{cases} if\ \ C_1 > C_2,\ then\ \ swap\ \ C_1\ \ and\ \ C_2; \\ if\ \ C_1 = C_2,\ then\ \ set\ \ C_2 = C_1\ +\ T_2, \end{cases} \tag{3.4}$$

   where T_2 is a pre-defined threshold.

4. Utilized the DCT coefficient C_w located at (2, 2) as mentioned before to embed a visible watermark pixel as follows:

$$\begin{cases} if\ \ w_{ij}\ \ is\ \ black,\ \ then\ \ set\ \ C_w = C_w - k; \\ otherwise,\ keep\ \ C_w\ \ unchanged, \end{cases} \tag{3.5}$$

   where k is a pre-defined constant.

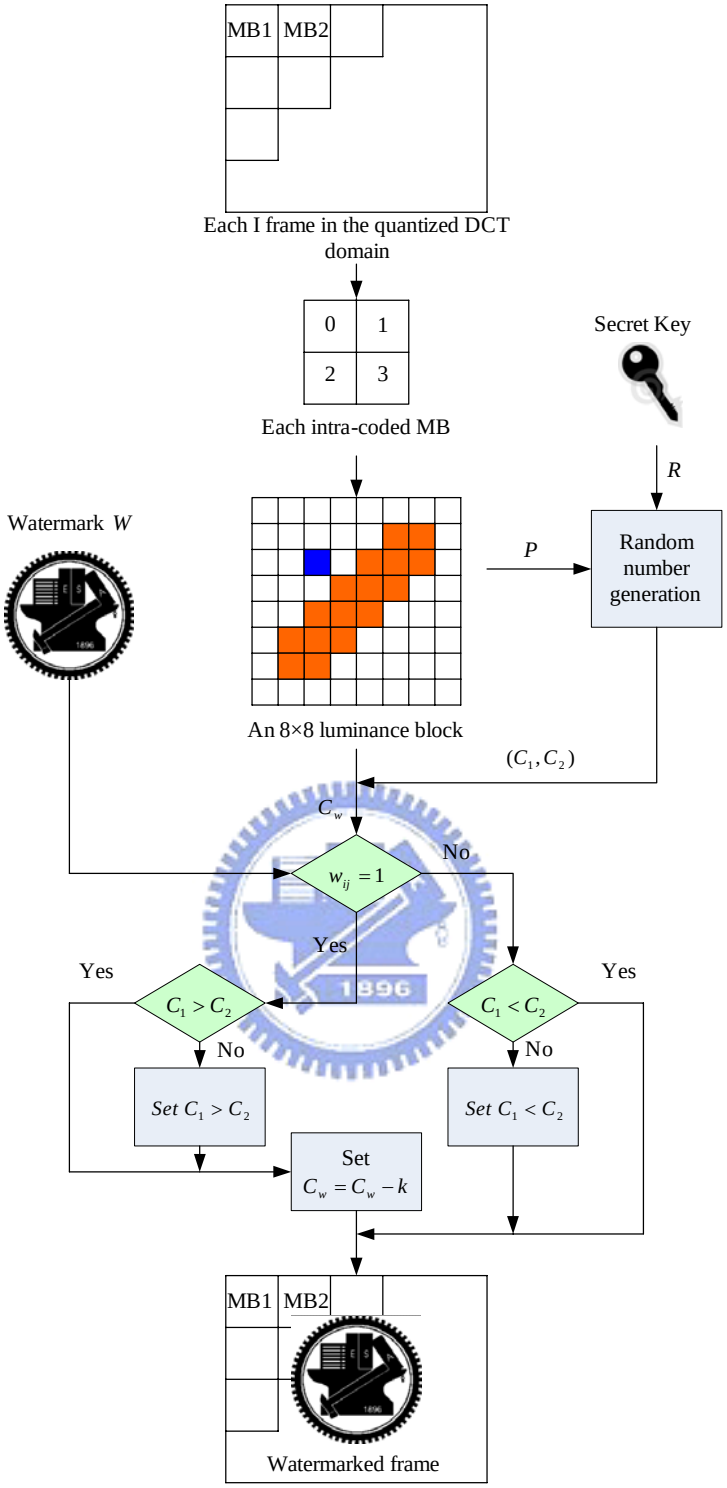


Figure 3.6 Flowchart of watermark embedding process for I frames

3.3.3 Process for Embedding Visible Watermarks in P and B Frames

There are two types of macroblocks which are used for embedding watermark

pixels in this study, namely, the intra-coded and inter-coded macroblocks. The embedding process of intra-coded macroblocks for P frames is the same as the method used for I frames. We focus the embedding process for inter-coded macroblocks in this section.

According to the basic concept of motion compensated prediction in the MPEG standard, P and B frames take I and P frames as forward or backward reference frames. In other words, most watermark pixels, but not all of them, embedded in I and P frames will appear in P and B frames. In order to embed the rest of watermark pixels, we have to introduce a number called the *coded block pattern* (CBP). A macroblock consists of six 8×8 blocks: four luminance blocks and two chrominance blocks. The CBP is a number composed of six bits; each bit being mapped to each of six 8×8 blocks, respectively, and represents whether the difference between the corresponding block and its reference block has been encoded into the bitstream. Only when a bit in the CBP equals 1 do we allow the corresponding 8×8 block to be watermarked; on the contrary, we do nothing. A flowchart of the embedding process for P and B frames is shown in Figure 3.7 and a corresponding detailed algorithm is described in the following.

Algorithm 3.3: Watermark embedding process for P and B frames.

Input: a P or B frame F in the quantized DCT-domain, a secret key R, and a binary watermark image W.

Output: a watermarked frame F' with a visible watermark W' similar to W.

Steps:

1. For each macroblock M of F, if the type of M is intra-coded, then perform the steps stated in Algorithm 3.2; otherwise, perform Step 2.

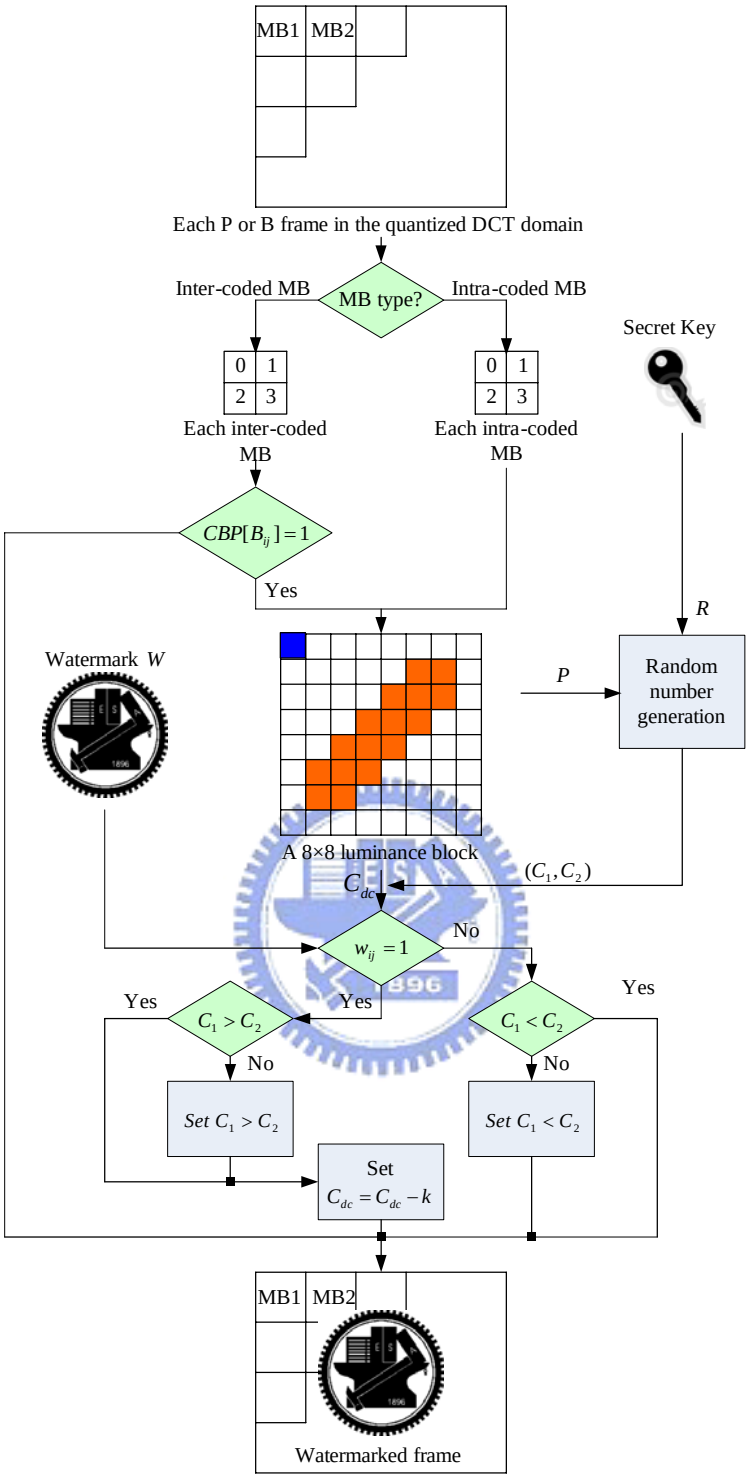


Figure 3.7 Flowchart of watermark embedding process for P or B frames

2. For each inter-coded macroblock *M* of *F*, perform the following operations:

   2.1 For each 8×8 luminance block B_{ij} of *M*, use $CBP[B_{ij}]$ to denote the bit in the CBP associated with the corresponding block B_{ij}.

2.2 If $CBP[B_{ij}] = 1$, then perform Step 2.3. Otherwise, skip to the next block B_{ij}.

2.3 Combine the position P of B_{ij} in F and the input secret key R to form a seed for a random number generation. Select one pair of coefficients (C_1, C_2) from the pre-defined eight pairs of coefficients according to the result of random number generation.

2.4 Embed the invisible signals, each of which represents whether the watermark pixel w_{ij} of W to be embedded into B_{ij} is black or white, by changing the relation between C_1 and C_2 to by the following rule:

(1) When w_{ij} is black:

$$\begin{cases} if \;\; C_1 < C_2, \; then \;\; swap \;\; C_1 \;\; and \;\; C_2; \\ if \;\; C_1 = C_2, \; then \;\; set \;\; C_1 = C_2 \; + \; T_2, \end{cases} \tag{3.6}$$

where T_2 is a pre-defined threshold.

(2) When w_{ij} is white:

$$\begin{cases} if \;\; C_1 > C_2, \; then \;\; swap \;\; C_1 \;\; and \;\; C_2; \\ if \;\; C_1 = C_2, \; then \;\; set \;\; C_2 = C_1 \; + \; T_2, \end{cases} \tag{3.7}$$

where T_2 is a pre-defined threshold.

2.5 Utilize the DC coefficient C_{dc} of B_{ij} to embed a visible watermark pixel as follows:

$$\begin{cases} if \;\; w_{ij} \;\; is \; black, then \; set \; C_{dc} = C_{dc} - k; \\ otherwise, keep \; C_{dc} \; unchanged, \end{cases} \tag{3.8}$$

where k is a pre-defined constant.

3.4 Recovery of Original Videos by Removing Visible Watermarks

The activation of the recovery process depends on whether the active player is able to get the secret key or not; in other words, not only the network must be available but also the video play count must be greater than zero. If no play count is left or the personal data does not match the right one stored in the server, the recovery process will not be performed and the video will be played with a visible watermark. The main task described in this section is to find out the watermarked blocks and to remove the watermark pixel by using the secret key received from the server. In Section 3.4.1, the process for extracting the personal data is described. The recovery process of I frames is described in Section 3.4.2 and the process for recovery of P and B frames is described in Section 3.4.3.

3.4.1 Process for Extraction of Personal Data

In this process, only the 8×8 luminance blocks used for hiding personal data bits are focused. Therefore, we have to extract the length of the personal data in binary form first because this length decides how many 8×8 blocks are used to extract the personal data bits. After completing the extraction process of the personal data, the result will be sent to the specific server automatically. A corresponding detailed algorithm is described in the following.

***Algorithm* 3.4**: The process for extraction of personal data.

Input: the first I frame *F*.

Output: the extracted personal data *D*

Steps:

1. For each 8×8 block used to hide the personal data, select two pre-defined DCT coefficients located at (6, 4) and (5, 5) as a pair (C_1, C_2) in the 8×8 matrix illustrated in Figure 3.3.
2. Extract a bit b_i as part of D according to the following rule:

$$\begin{cases} if \;\; C_1 > C_2, \;\; then \;\; set \;\; b_i = 0; \\ if \;\; C_2 > C_1, \;\; then \;\; set \;\; b_i = 1. \end{cases} \tag{3.9}$$

3.4.2 Process for Recovery of I Frames

After the personal data are verified by the server and the play count is large enough to be deducted, the active player will receive the secret key and perform the recovery process. For each 8×8 luminance block in the I frame, we randomly select a pair of DCT coefficients (C_1, C_2) from the pre-defined pairs and check the relation between C_1 and C_2 to decide whether this block is used to embed a black or white watermark pixel. If the relation between C_1 and C_2 results in a black watermark pixel, the embedded watermark pixel in this block will be removed; otherwise, we do nothing to this block. A flowchart of the watermark removal process for I frames is shown in Figure 3.8 and a detailed algorithm is described in the following.

***Algorithm* 3.5**: The process for recovery of I frames.

Input: a watermarked I frame F', a secret key R.

Output: a recovered I frame F.

Steps:

1. For each 8×8 block B_{ij} of F', combine the input secret key R and the position P of B_{ij} to form a seed for a random number generation.
2. Select one pair of coefficients (C_1, C_2) from the pre-defined eight pairs of coefficients according to the result of random number generation.

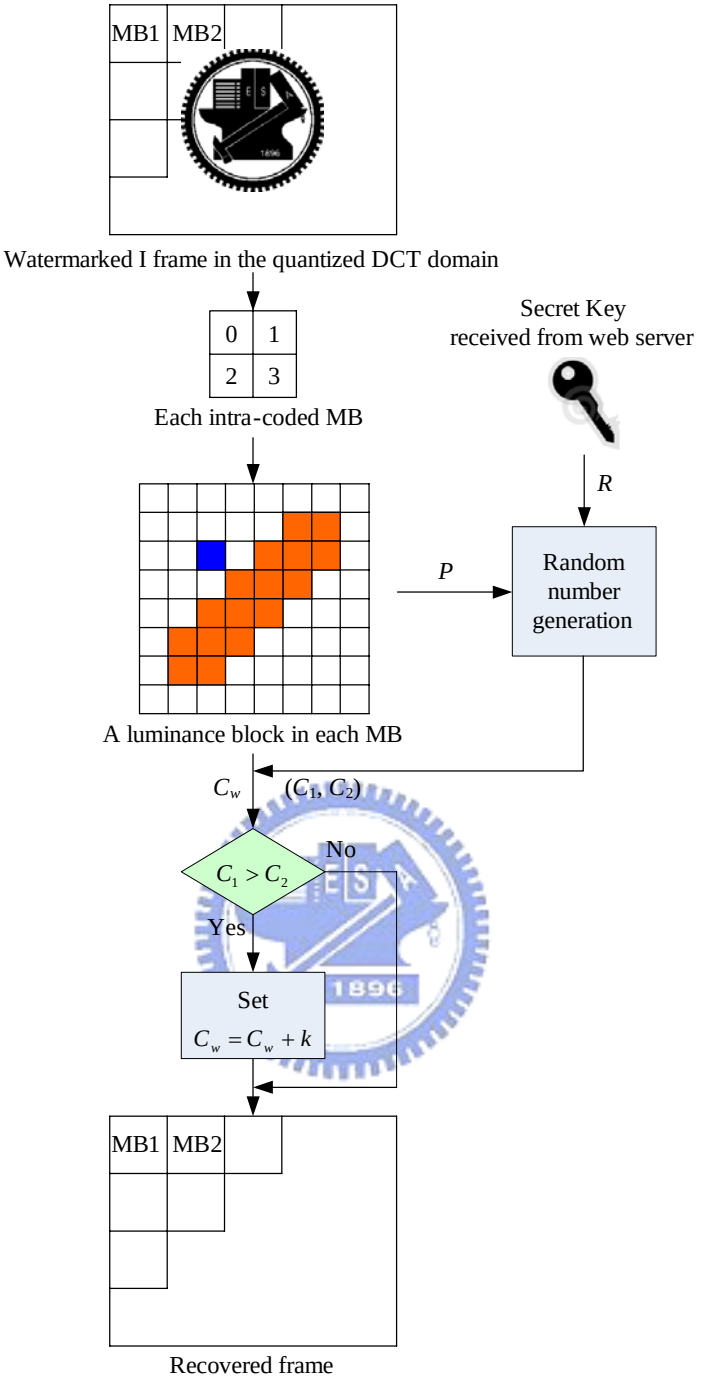


Figure 3.8 Flowchart of the watermark removal process in I frames.

3. Check the relation between C_1 and C_2 to determine whether the watermark pixel w_{ij} embedded in B_{ij} is black or white by the following rule:

$$\begin{cases} if \;\; C_1 > C_2, \;\; then \;\; w_{ij} \;\; is \;\; black; \\ if \;\; C_2 > C_1, \;\; then \;\; w_{ij} \;\; is \;\; white. \end{cases} \tag{3.10}$$

4. Modify the pre-defined DCT coefficient C_w located at (2, 2) to remove the

watermark pixel embedded in B_{ij} by the following rule:

$$\begin{cases} if \;\; w_{ij} \;\; is \;\; black, \;\; then \;\; set \;\; C_w = C_w + k; \\ otherwise, \;\; keep \;\; C_w \;\; unchange, \end{cases} \tag{3.11}$$

where k is a pre-defined constant.

3.4.3 Process for Recovery of P and B Frames

We use the secret key received from the server to continue the recovery process for P and B frames. The watermark removal process for intra-coded macroblocks is similar to the method used for I frames. Moreover, we utilize the CBP to check whether each 8×8 block has been used to embed a watermark pixel during the watermark removal process for inter-coded macroblocks. A flowchart of the watermark removal process for P and B frames is shown in Figure 3.9 and a detailed algorithm is described in the following.

***Algorithm* 3.6**: The process for recovery of P and B frames.

Input: a watermarked P or B frame F', and a secret key R.

Output: a recovered P or B frame F .

Steps:

1. For each macroblock M of F', if the type of M is intra-coded, then perform the steps stated in Algorithm 3.5; otherwise, perform Step 2.
2. For each inter-coded macroblock M of F', perform the following operations.
   2.1 For each 8×8 luminance block B_{ij} of M, use $CBP[B_{ij}]$ to denote the bit in the CBP associated with the corresponding block B_{ij}.
   2.2 If $CBP[B_{ij}] = 1$, then perform Step 2.3. Otherwise, skip to the next block B_{ij}.

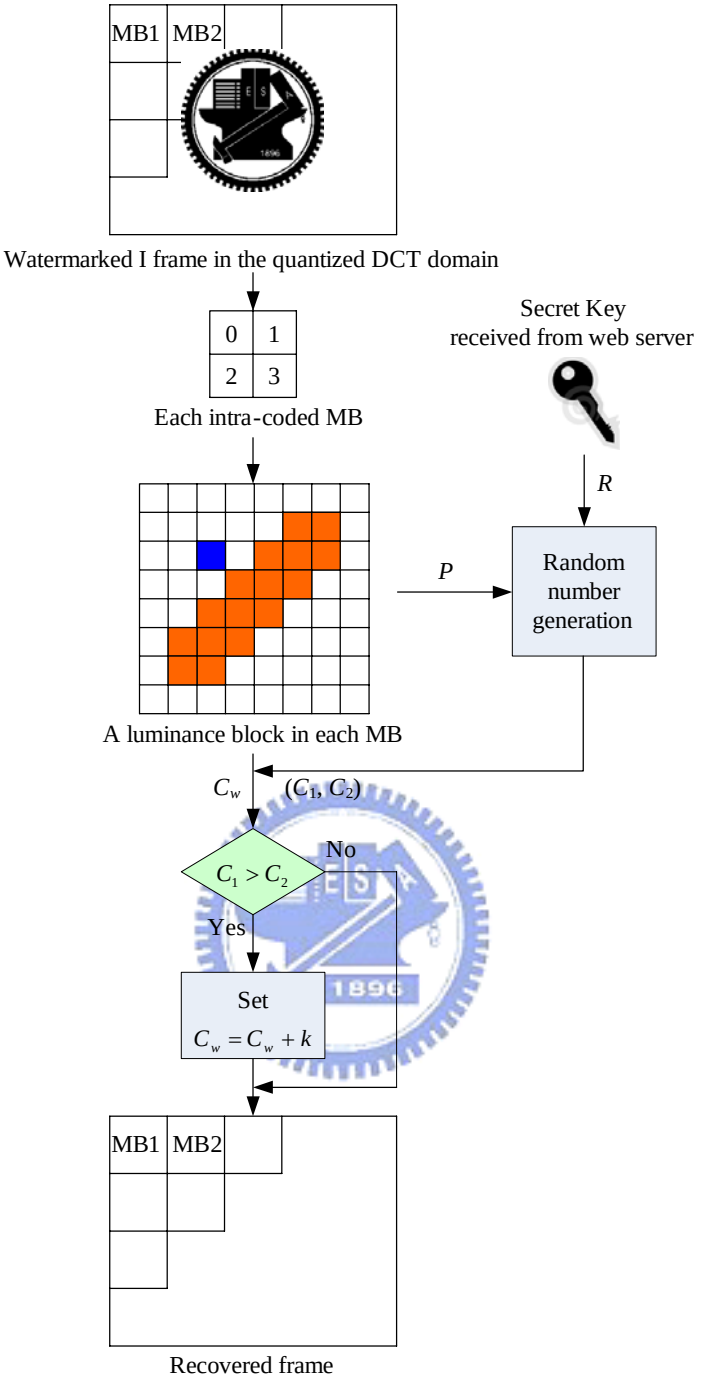


Figure 3.9 Flowchart of the watermark removal process in P or B frames.

2.3 Combine the position P of B_{ij} in F' and the input secret key R to form a seed for a random number generation. Select one pair of coefficients (C_1, C_2) from the pre-defined eight pairs of coefficients according to the result of random number generation.

2.4 Check the relation between C_1 and C_2 to determine whether the

watermark pixel w_{ij} embedded in B_{ij} is black or white by the following rule:

$$\begin{cases} if \;\; C_1 > C_2, \;\; then \;\; w_{ij} \;\; is \;\; black; \\ if \;\; C_2 > C_1, \; then \;\; w_{ij} \;\; is \;\; white. \end{cases} \tag{3.12}$$

2.5 Modify the DC coefficient C_{dc} to remove the watermark pixel embedded in B_{ij} by the following rule:

$$\begin{cases} if \;\; w_{ij} \; is \; black, then \; set \; C_{dc} = C_{dc} + k; \\ otherwise, keep \; C_{dc} \; unchanged, \end{cases} \tag{3.13}$$

where k is a pre-defined constant.

3.5 Experimental Results

In our experiments, an MPEG-4 video with frame size 320×240 was used as the input to embed a watermark image with size 30×30. The binary watermark image is shown in Figure 3.10. Six frames of the original video are shown in Figure . The corresponding six frames of the watermarked video are shown in Figure 3.. The corresponding six frames of the recovered video are shown in Figure . The proposed active player presented in Figure 3.14(a) shows a recovered video whose play count is greater than zero. If the video play count has run out or the active player encounters network problems while connecting to the server for checking the video play count, a pop-up message dialog shown in Figure 3.14(b) and Figure 3.14(d) and a watermarked video shown in Figure 3.14(c) and Figure 3.14(e) will be displayed to the user. Figure 3.14(f) illustrates the result of trying to play the downloaded video with an improper video player.

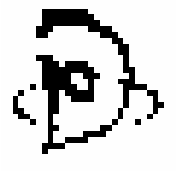
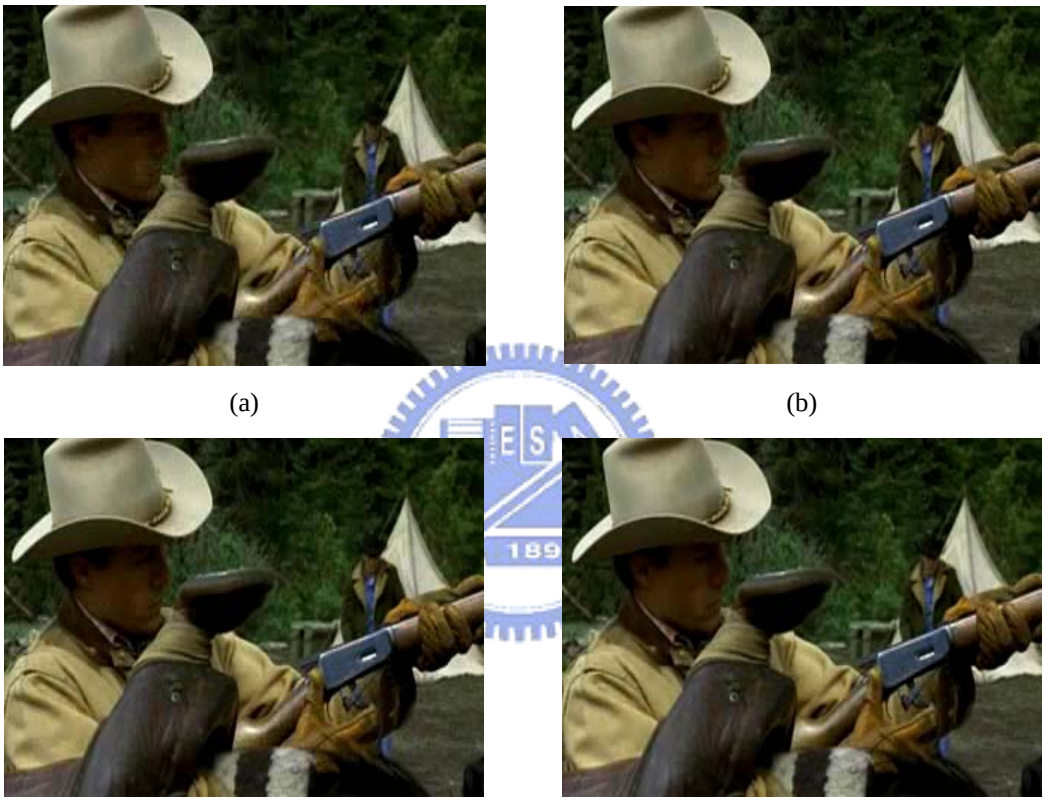


Figure 3.10 A watermark binary image with size 30×30



(a) (b)

(c) (d)

Figure 3.11 Six frames of the original video. (a) The first frame (I frame). (b) The second frame (B frame). (c) The third frame (P frame). (d) The 4th frame (B frame). (e) The 5th frame (P frame). (f) The 6th frame (B frame). (continued)

(e) (f)

Figure 3.11 Six frames of the original video. (a) The first frame (I frame). (b) The second frame (B frame). (c) The third frame (P frame). (d) The 4th frame (B frame). (e) The 5th frame (P frame). (f) The 6th frame (B frame). (continued)

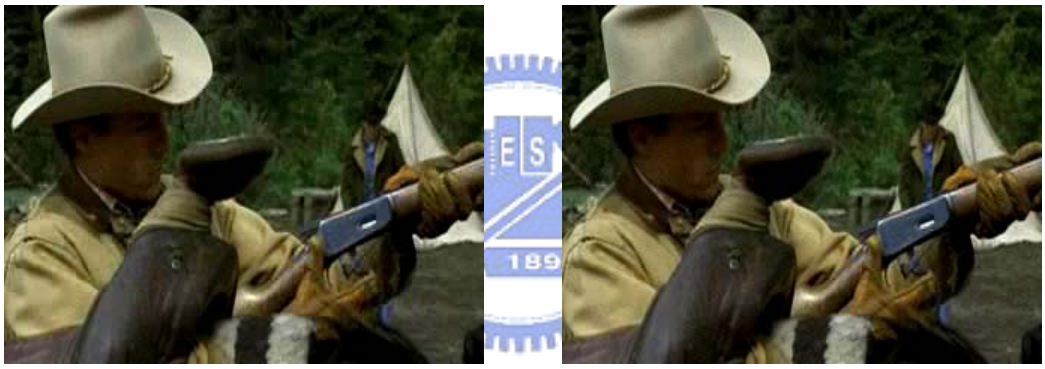
(a) (b)

(c) (d)

Figure 3.12 Six frames of the watermarked video. (a) The first frame (I frame). (b) The second frame (B frame). (c) The third frame (P frame). (d) The 4th frame (B frame). (e) The 5th frame (P frame). (f) The 6th frame (B frame). (continued)

(e) (f)

Figure 3.12 Six frames of the watermarked video. (a) The first frame (I frame). (b) The second frame (B frame). (c) The third frame (P frame). (d) The 4th frame (B frame). (e) The 5th frame (P frame). (f) The 6th frame (B frame). (continued)



(a) (b)

(c) (d)

Figure 3.13 Six frames of the recovered video. (a) The first frame (I frame). (b) The second frame (B frame). (c) The third frame (P frame). (d) The 4th frame (B frame). (e) The 5th frame (P frame). (f) The 6th frame (B frame). (continued)

(e) (f)

Figure 3.13 Six frames of the recovered video. (a) The first frame (I frame). (b) The second frame (B frame). (c) The third frame (P frame). (d) The 4th frame (B frame). (e) The 5th frame (P frame). (f) The 6th frame (B frame). (continued)

(a)

Figure 3.14 (a) A recovered video with the play count greater than zero. (b) A pop-up message dialog saying that the play count of the video has run out. (c) A watermarked video with the play count equals zero. (d) A pop-up message dialog saying that the network is not available. (e) A watermarked video resulting from some network problems. (f) A watermarked video played on an improper video player. (continued)

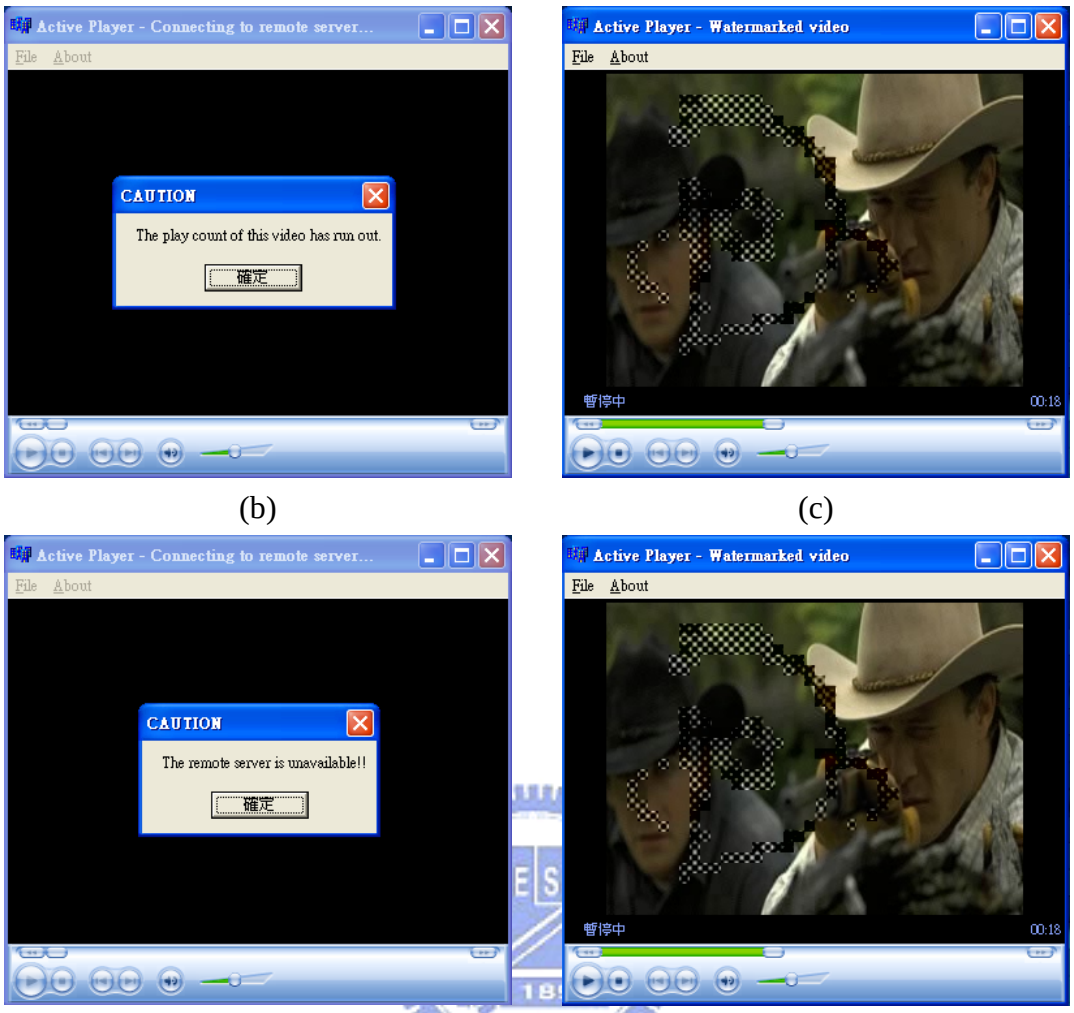
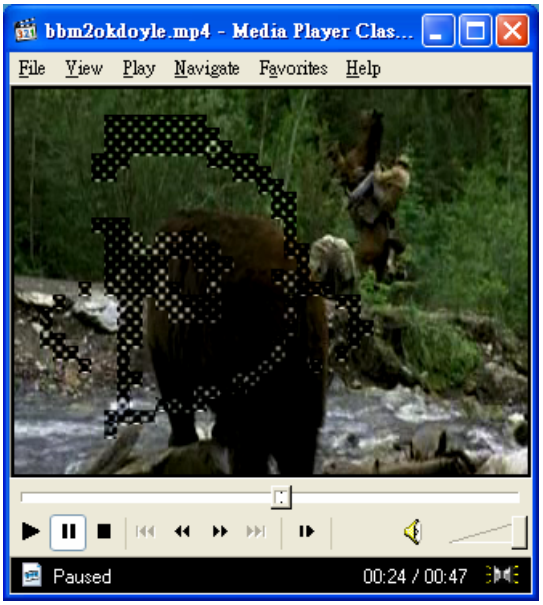


Figure 3.14 (a) A recovered video with the play count greater than zero. (b) A pop-up message dialog saying that the play count of the video has run out. (c) A watermarked video with the play count equals zero. (d) A pop-up message dialog saying that the network is not available. (e) A watermarked video resulting from some network problems. (f) A watermarked video played on an improper video player. (continued)



(f)

Figure 3.14 (a) A recovered video with the play count greater than zero. (b) A pop-up message dialog saying that the play count of the video has run out. (c) A watermarked video with the play count equals zero. (d) A pop-up message dialog saying that the network is not available. (e) A watermarked video resulting from some network problems. (f) A watermarked video played on an improper video player. (continued)

3.6 Discussions and Summary

In this chapter, we have proposed a scheme to protect the copyright and ownership of downloaded videos by briefly communicating with the server and limiting the play counts of these videos through the use of an active video player. The embedded visible watermark which is different from traditional ones not only claims the copyright but also presents a twinkling effect to prevent users from enjoying the movies without paying the fee. In addition, the proposed system can be applied to rental applications as an online movie rental system.