Active Copyright Protection of Html Files by A Watermarking Technique

Yuei-Cheng Chuang (莊岳城)[1] and Wen-Hsiang Tsai (蔡文祥)[1, 2]

[1]Department of Computer & Information Science
National Chiao Tung University, Hsinchu, Taiwan 300
[2]Department of Computer Science & Information Engineering
Asia University, Taichung, Taiwan 413
E-mail: {gis92615, whtsai}@cis.nctu.edu.tw

ABSTRACT

With the popularity of computer networks, more and more information are displayed on web pages. Because illegal users may download html files for misuses, authors may want to protect the copyright of them. An active digital watermarking method for this purpose is proposed. An active agent is embedded in the html file to transform a controllable invisible watermark into a visible one when a request of downloading the html file is issued by an illicit user. Experimental results show the feasibility and practicability of the proposed method.

Keywords: active watermarking, html file, copyright protection, active agent, invisible watermark.

1. Introduction

Recently, more and more digital data are transmitted on the public network. People can search information by browsing web pages on the Internet. Illegal users might copy information from web pages by downloading the html files of the web pages. So protection of the copyright of the web page, or alternatively, of the html file is very important. In this study, we propose an *active watermarking* method to detect downloading actions requested by users and transform invisible watermarks hidden in the html files into visible ones *actively* by an active agent.

Generally speaking, an active agent is an active data stream executable to perform specific tasks actively by itself, such as showing the ownership of the cover media, scrambling the cover media when authenticity checks fail, etc. It makes the cover media look *alive*. The html format studied in this research supports the script language of JavaScript. By this language, many functions like creating additional objects in html files and manipulating the properties of objects, etc., can be defined. In other words, this script language can perform specific tasks actively by itself. So programs of this script language can be regarded as active agents.

Many information hidng techniques have been proposed for copyright protection of images and files [1-6]. Yu, et al. [1] proposed a method for hiding an active agent in a cover media without using other media. But this method costs a lot of the information hiding capacity. Chang and Tsai [2] proposed a method for hiding a watermark in an html file for copyright protection, but the method can not protect the html file *actively*. Lo and Tsai [3] proposed an improved system of Chang and Tsai [2], whose focus was on the vedio.

The active copyright protection techniques proposed in this study includes two functions. One is hiding a controllable watermark into an html file. Unless a correct key is given, the html file can not be recovered and displayed. The other function is to determine whether the protected html file is used legally or not. If legal, the key will be transmitted from the server site to the html file at the client site, and the html file will then transform the controllable visible watermark into an invisible one and recover itself to show the web page which it describes.

In Section 2, we will describe the proposed controllable watermark hiding technique for html files. In Section 3, we will describe the proposed technique for verifying the legal status of a suspicious html file by matching the updated parameters in the server and those of the html file. And in Section 4 some experimental results will be shown. Conclusions are made in the last section.

2. Proposed Controllable Watermark Hiding Techniques For Html Files

Fig. 1 illustrates the proposed process of embedding a controllable watermark into an html file. We divide the text of the html file into multiple text blocks, and assign each text block an index number. Fig. 2 illustrates this process of html dividing and index assigning. We then record the indexes of the text blocks and randomize the order of the text blocks for security promotion. We also transform a given logo image into multiple image clips for hiding the index numbers of the blocks. Then we embed the image clips into the html file to be protected. Finally, we embed the extraction and recovery codes written as JavaScript into the html file. Unless a correct key is given, the html file can not be recovered for browsing.

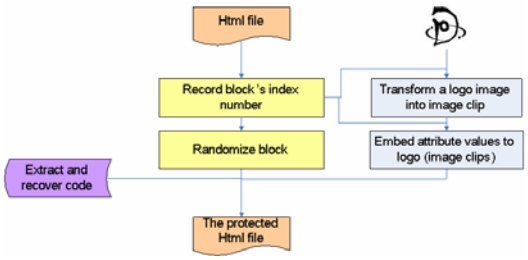


Fig. 1 Proposed active copyright protection procedure.

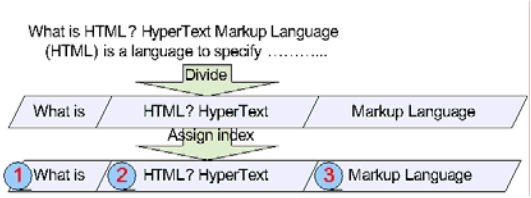


Fig. 2 Process of html dividing and index assigning.

The reason for randomizing the properties of the texts in the html file is described in Section 2.1. In Section 2.2, the method for hiding information by using the relations of image clips will be described. And why and how to transform a logo image into images is described in Section 2.3. In Section 2.4, a detailed algorithm for controllable watermark embedding will be described. Finally, in Section 2.5, the detailed algorithm for recovering information for watermark extraction will be described.

2.1 Security Promotion by Randomizing Movie Clip Attributes in Html Files

The content of a protected html file includes two parts. The first part, named the *display part*, is used for displaying the scenario of the file. The second part, named the *protection part*, is used for protecting the file. If an attacker wants to destroy the protection function embedded in the html file, he/she can remove or modify the protection part. How can we solve this problem? An answer is as follows: if the protection part includes the information which is necessary for displaying the html file, then the part will not be removed or modified, because removing or modifying it will destroy the relevant information included in it, making it impossible to display the html file normally.

In order to achieve this effect, our idea is to record the indexes of the text blocks, randomize the text blocks in the html file, and use the indexes to record the randomized block position relations. Fig. 3 shows an example, in which the sentence "What is HTML? Hyper Text Markup Language" is divided into three text blocks. They are "What is," "HTML? Hyper Text," and "Markup Language." They are assigned the indexes "1," "2," and "3," respectively. We randomize the order of the blocks using the indexes, and the sentence becomes "HTML? Hyper Text Markup Language What is," in which the text "HTML? Hyper Text" is placed at the first position of the sentence, and so on. We record the position relations between the blocks using the indexes so that the randomized text blocks can be restored to compose the original sentence. Such relations are called *recovery information* in this study.

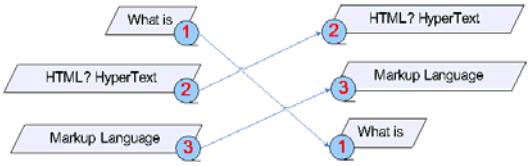


Fig. 3 An example of randomizing the order of the positions of text blocks using index numbers.

And now the question is: how can we attach the recovery information to the protection part of the html file? The technique we propose in this study is to embed a visible watermark in the html file, and hide the recovery information with a user key in the watermark. Because the content of the html file is already destroyed by randomization, it is also necessary to embed certain codes to recover the html file for legal users. The codes are designed to extract the recovery information from the visible watermark and to restore the information into the text blocks.

Because the recovery information is embedded in the watermark, if someone wants to remove or modify the watermark, the embedded recovery information may be destroyed. On the other hand, because the extraction and recovery codes are used for recovery information extraction and restoring, if someone removes or modifies the codes, then the information recovery

work can not be performed normally by the damaged extraction and recovery codes.

2.2 Information Hiding by Image Clip Relations

In this study, we utilize the script language to program the active agent. Because of the limitation of the script language, we must hide information at places where the attributes of objects can be controlled and detected by the script language. In this section, we describe how we use the relation between the properties of two images for hiding the recovery information. Some properties of images can be read and written, such as position, alpha, scale. So we can embed the recovery information by using the relations among image properties. For example, we may use the scales of images for information hiding. Suppose that we have two images, ic_1 and ic_2. The scale of ic_1 is is_1, and that of ic_2 is is_2. We hide a binary bit d by the following rule in this study:

if $d = 1$, *let* $is_1 > is_2$; otherwise, *let* $is_1 < is_2$.

We can extract the hidden binary data from a group of image clips easily. Specifically, we extract a binary bit α by the following rule:

if $is_1 > is_2$, set $\alpha = 1$; otherwise, *set* $\alpha = 0$.

Hiding and extracting binary data by the use of other image properties may be conducted in similar ways.

2.3 Transforming Logo Image into Image Clips and Embedding of Recovery Information

In this study, we embed a visible watermark in an html file for declaring the ownership of the html file. The watermark will be displayed in the web page created by the file. How can we do this? A simple way is just to place a logo image in the html file. But this method is impractical because the logo image has no ability to protect itself, and an illicit user can still copy the text from the html file directly. Another way is to place a logo image in the html file, and modify the properties of the logo image to make the image to appear over a text of the html file. That is, the text will be covered by the logo image so that an illicit user can not see the content of the text. But unfortunately, the format of html file is open in nature, and so a user can download the protected html file easily and remove the logo image in the html file, so that the text will be revealed. A third and practical way, which is devised in this study, is to randomize the text of the html file and embed the recovery information described in Section 2.1 into the logo image, so that the logo image will not be removed because the recovery information is indispensable for displaying the html file as a web page.

Now how can we embed the recovery information in the logo image effectively? Although many information hiding techniques have been proposed, they are not suitable for hiding the recovery information mentioned previously. The reason is that most of these methods hide information by the use of the relations among image pixel values, and unfortunately the script language can not read image pixel values. To solve this problem, in this study we propose the method of *transforming the logo image into many image clips*, and use the relations between the clips to hide information. Fig. 4 shows an example of a logo image transformed into many image clips. A detailed algorithm of the transformation process is described as follows.

(a) (b)

Fig. 4 A logo Image and its image clips. (a) Logo image. (b) Image Clips.

***Algorithm* 1:** transforming a logo image into image clips and embedding the recovery information.

***Input*:** a user key K; a binary logo image W; recovery information R; and an html file H.

***Output*:** a set of image clips W' with R embedded in it.

***Steps*:**

Step 1. Calculate the number N of image clips which will be used for hiding R.

Step 2. Resize W according to the dimension of H, resulting in W'.

Step 3. Sample the pixels of W' according to N and record the coordinates of the sampled pixels into a set C.

Step 4. Create a set of image clips IC and assign to each clip in IC the coordinates of a corresponding pixel in C.

Step 5. Use the key K to disarrange the content of R randomly, resulting in a new date set R'.

Step 6. Use the hiding method described in Section 2.2 to hide R' in IC.

2.4 Proposed Controllable Watermark Embedding Process

Based on Algorithm 1, we can now describe a complete algorithm to implement the proposed idea of controllable watermark embedding.

***Algorithm* 2:** controllable watermark embedding.
***Input*:** an html file H; a logo image W; and a secret key K.
***Output*:** a protected html file.
***Steps*:**
Step 1. Divide the text of H into a set of text blocks.
Step 2. Assign an index to each text block.
Step 3. Randomize the positions of the text blocks.
Step 4. Record the recovery information R including the position relations of the blocks in terms of the block indexes.
Step 5. Take R, H, W, and K as input, and perform Algorithm 1 to transform the logo image W into a group of image clips, yielding a set W' of the image clips of the logo image W.
Step 6. Embed W' into H.
Step 7. Design and embed corresponding extraction and recovery codes into H.

2.5 Proposed Html File Recovery Process

In this section, the proposed protected html file recovery scheme will be described. Recall that after performing the controllable watermark embedding process described by Algorithm 2 to an html file, the html file becomes a protected one. The Javascript codes embedded into the protected html file in Step 7 of Algorithm 2 has been given an ability to recover the text blocks. The detailed algorithm for recovering the protected html file is described as follows.

***Algorithm* 3:** extraction of recovery information and recovering of a protected html file.
***Input*:** a protected html file H and a secret key K.
***Output*:** an html file that can be displayed normally.
***Steps*:**
Step 1. Extract the recovery information R embedded in the image clips of the logo image with K.
Step 2. Use R to restore the randomized text blocks by using the relations of the text blocks kept in R.
Step 3. Transform the logo image from visible to invisible.

3. Activeness Creation by Matching Updated Parameters in Server And HTML Files

In this section, we will describe the way we propose to determine whether a protected html files are used legally or not. We will embed parameters in the protected html file for checking the status of legal usage. We also embed a "send back" program in the file for sending parameters from the client site to the server site. There is a decision program at the sever site which receives the messages from the client and determines whether the html file is played legally or not. If legal, the key will be transmitted from the server to the html file at the client site, and the html file will transform the controllable visible watermark into an invisible one for the user to see that content of the html file in the form of a browsed web page. The structure of the server site is shown in Fig. 5.

Our method is to use two parameters to decide whether the html file is used legally or not. An update program in the server will update two parameters (named Parameter 1 and Parameter 2) in the protected html file *periodically*. The frequencies of updating the two parameters are the same, but the start points of updating these two parameters are different. The decision scheme is described in the following.

First, the two parameters are sent back to the server from the client site using the "send back" program in the html file. The decision program in the server compares each of the two parameters extracted from the protected html file at the server site with the parameters received from the client site. Unless these two pairs of parameters are *both* different, the html file will be displayed at the client side.

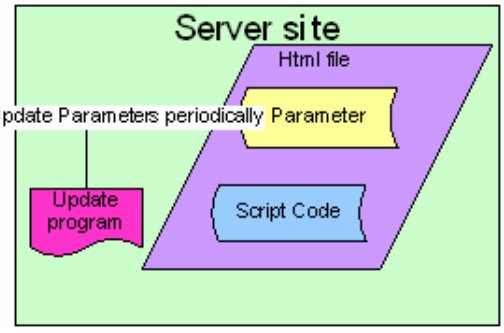


Fig. 5 Server site structure.

Why does the above-mentioned scheme work for the purpose of protecting legal usage of the html file? First, if an html is downloaded illegally into a client site for a sufficiently long period of time, the two parameters in the file at

the client site *will not* be updated. But those in the html file in the server *will*. Therefore, when the file at the client site is executed off-line, the parameters sent back to the server by the "send back" program will both be out of date and will be checked to be so by the decision program at the server site, thus achieving the function of protection (i.e., keeping the visible watermark on the text of the web page).

On the other hand, when the html file is executed on-line normally (i.e., when the corresponding web page is browsed on-line) by a legal user, because at any time either of the two parameters will be updated, the comparison will be successful so that the file can be displayed normally with the visible watermark being removed. The reason why we use two parameters instead of just a single one is that a legal user might unluckily browse the web page (i.e., execute the html file) at a moment when the single parameter is not updated, and in that case, the comparison will not pass and the visible watermark will not be removed.

4. Experimental Results

In our experiments, two applications for active copyright protection of html files were tested. One is transformation of invisible watermarks into visible ones actively, and the other is an inverse process.

An example of the first experiment shown here. A normal html file displayed as a web page is shown in Fig. 6(a). Assume that a user wants to steal the html file from the server site. The result is that the active agent will transform the controllable invisible watermark into a visible one. And a dialog will pop up, asking a user key. This phenomenon is shown in Fig. 6(b).

In the second experiment, the html file is protected by a visible watermark initially. An example is shown in Fig. 6(c). When a user wants to see a clear html file off-line, an authentic key should be provided to the html file, and the html file will then recover for browsing. Fig. 6(d) shows the normal html file being played off-line. If the given key is incorrect, the html file will recover erroneously. Fig. 6(e) shows an erroneous off-line display of the html file. Note the disorder of the displayed blocks of the text content.

5. Conclusions and Discussions

In this study, we have proposed an active watermarking method for copyright protection of html files. An active agent, implemented by the use of Javascript language programs, is embedded in the html file to transform a controllable invisible watermark into a visible one when a request of downloading the html file is issued by an illicit user. Experimental results show the feasibility of the proposed method.

The active copyright protection scheme proposed in this study is only a concept. Many different extensions based on the concept can be carried out easily. For example, we can randomize other data in the html file, such as images, hyperlinks, and so on. Different randomization methods have different functions of protections.

6. Acknowledgement

This work was supported partially by the NSC project Advanced Technologies and Applications for Next Generation Information Networks (II) – Sub-project 5: Network Security with Project No. NSC93-2752-E-009-006-PAE.

7. References

[1] H. H. Yu, et al., "Smart Media: empower media with active data hiding," *Proceedings of 6th International Computer Science Conference on Active Media Technology*, Hong Kong, China, Vol. 2252, pp. 5-16, December 19-29, 2001.

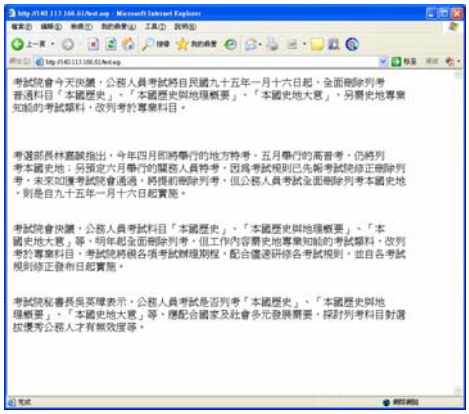
[2] Y. H. Chang and W. H. Tsai, "A steganographic method for copyright protection of HTML documents," *Proceedings of 2003 National Computer Symposium,* Taichung, Taiwan, Republic of China, Dec. 2003.

[3] N. K. Lo and W. H. Tsai, "A study on active information hiding and applications," *Technical Report*, Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, June, 2004.

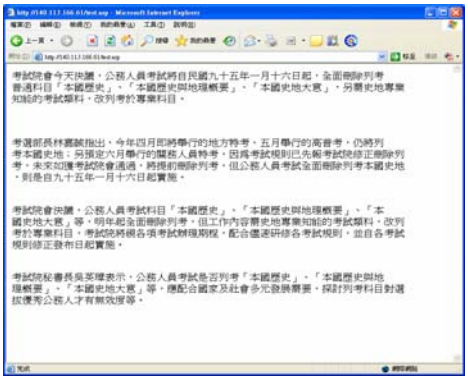
[4] D. C. Wu and W. H. Tsai, "Spatial-domain image hiding using an image differencing," *IEE Proceedings -Vision, Image and Signal Processing*, Vol. 147, No. 1, pp. 29-37, 2000.

[5] C. H. Tzeng and W. H. Tsai, "A new approach to authentication of binary images for multimedia communication with distortion reduction and security enhancement," *IEEE Communications Letters*, Vol. 7, No. 9, pp. 443-445, 2003.

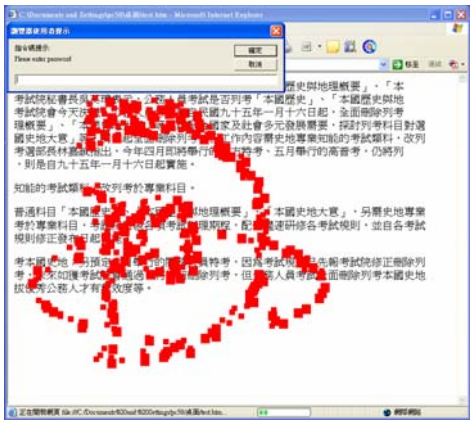
[6] C. H. Tzeng, Z. F. Yang, and W. H. Tsai, "Adaptive data hiding in palette images by color ordering and mapping with security protection," *IEEE Transactions on Communications*, Vol. 52, No. 4, pp. 791-800, 2004.



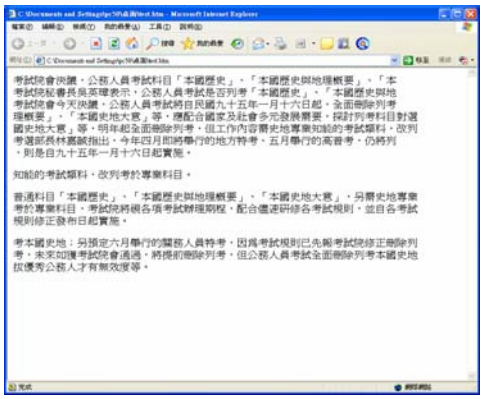
(a)



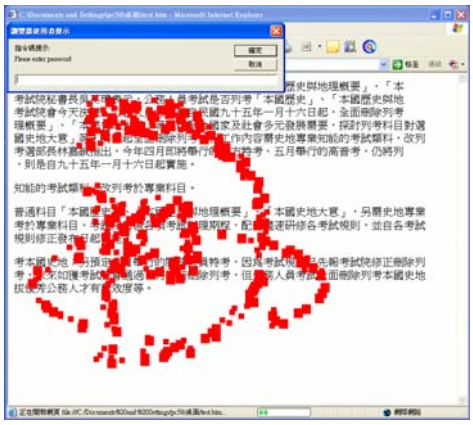
(d)



(b)



(e)



(c)

Fig. 6 Experimental results. (a) A clear html file with invisible watermarks displayed on-line. (b) A html file with visible watermarks and a pop-up input area. (c) A protected html file suffering from illegal downloading. (d) A clear html file with an invisible watermark after an authentic key was provided. (e) An erroneous html file display after an incorrect key was provided.