Journal of Network Intelligence ©2023 ISSN 2414-8105 (Online)
Taiwan Ubiquitous Information Volume **8**, Number **1**, February **2023**

Multipurpose Multilevel Multichannel Information Hiding for Web Pages

Lei Li, Hong-Jun Zhang
Command and Control Engineering College
Army Engineering University
Nanjing 210007, P. R. China
605531524@qq.com

Zhe-Ming Lu*
School of Aeronautics and Astronautics
Zhejiang University
Hangzhou, 310027, China
zheminglu@zju.edu.cn

*Corresponding author: Zhe-Ming Lu

Received September 1, 2022, revised October 17, 2022, accepted December 6, 2022.

Abstract. *Information hiding technology is a branch in the field of computer security technology. This technology is relatively mature in multimedia fields such as images, videos, audios, etc., and a large number of literatures have expounded related technologies. The file structure of the web page is composed of ordinary text files and various tags, and there is not much redundant information in this file structure, which makes it difficult to hide information into the web page file. The current research on web page information hiding technology is mainly based on the high error tolerance of HTML syntax and information hiding technology for web page tags. Most of these methods have the characteristics of insufficient watermark embedding capacity and poor robustness. In this paper, we propose a Multipurpose Multilevel Multichannel Information Hiding framework for web pages. In our framework, we design seven embedding algorithms for Web pages and we can achieve multiple purposes by using four-level independent watermark embedding in multiple channels of a Web page. Experimental results demonstrate the effectiveness of the proposed seven methods.*

Keywords: Information Hiding, Web watermarking, Multilevel watermarking, Multipurpose watermarking, Multichannel watermarking.

1. **Introduction.** With the popularization and development of information technology and the Internet, websites and web pages have been widely used, which brings great convenience to the production, operation and management of government departments and enterprises and institutions, as well as people's work and life. The Internet has become an indispensable tool for people to obtain information and exchange information. People can publish their work through the Internet, convey important information, etc. At the same time, the illegal copying of web pages and the counterfeiting of web pages are becoming more and more serious, which is an urgent problem to be solved. Using existing technologies, the web page is easy to be tampered with, and it is difficult to judge whether the web page is tampered with by the human eye. Authentication of the authenticity of web pages and tamper detection and positioning are becoming increasingly important. The traditional text summarization technology based on the hash function is

to calculate the message authentication code by hashing the source file of the web page, append the message authentication code to the pre-protected web page file, extract the message authentication code by pre-detecting the web page, and compare the obtained authentication code with the message authentication code generated by the pre-detection web page to determine whether the web page has been tampered with. However, in practical applications, this method often has many drawbacks, such as increasing the size of the file, and the message authentication code is easy to find.

Web page watermarking is a technology of web page protection that has appeared in recent years. It embeds copyright identification information or identity authentication information (watermark) in web pages in a certain way to prove the copyright attribution of web pages and identify the authenticity of web pages. In addition, web page watermarking technology can also be used to prevent web page tampering, hide and transmit secret information in web pages and so on. For example, we can embed the message authentication code directly into the web page itself by adding a space at the end of the line, upper-lower coding(ULC), or changing the position of the tag's attributes, it does not change the function of the page and does not increase the size of the file but it is easy to embed and extract. When using a digital watermark to certify the authenticity of a web page, it is usually a message authentication code or digital signature (called authentication information) of the web page data that is directly embedded in the web page.

Compared with the traditional research on image, video and audio watermarking, the research on web page watermarking is still very few. The structure of web pages is different from images and audio files, and the commonly used image and audio watermarking techniques are not suitable for web page watermarking. The HTML code representing the web page has a simple structure and less redundancy, and thus it is difficult to embed a watermark in the web page. Existing web page watermarking methods are mainly based on the principle that HTML language is not sensitive to the change of some information, and the embedding of watermark information is realized by changing these insensitive information. They are information hiding schemes that are mainly based on the high error tolerance of HTML syntax and web page tags. Typical methods are:

(1) Using the structural features of the HTML language to embed an invisible watermark into a web page [1]. Based on predefined semantic and syntactic rules, web text content is parsed to generate a watermark that goes through an encrypted cycle using a certain HASH algorithm to generate a hash which is replaced with invisible control characters to generate on an invisible watermark set.

(2) Embedding watermarks based on inserting invisible characters. Invisible characters, such as space and tab characters, can be embedded into the end of words or lines. These characters do not change the displaying result of web pages when they are browsed. The space and tab can denote the watermark bits '0' and '1' respectively. Although this scheme is implemented easily, the secret information can be easily found out by analyzing the source code of HTML since this method increases the file size.

(3) Modifying the case of tag names and attribute names [2]. This scheme chooses the double-quotes parts of HTML tags and the contents parts of HTML for protection, and use m-sequence to generate the watermark of web page. While embedding the watermark, this method only chooses the non-double-quotes parts of HTML tags and utilizes the case-insensitiveness of HTML tags.

(4) Adjusting the number of spaces between tag names and tag attributes to hide information [3], because there are many symbols between symbols. Each space will be treated as a single space, or insert a space to the left of the symbol '>' to hide the information, one or more spaces of the symbol '>' will be ignored by the browser, or

use the attribute assignment sign '=' in the tag to left and right to add spaces to hide information [4].

(5) Defining non-existing tags [5]. The method in [5] is a dual watermarking scheme based on threshold cryptography. This scheme consists of watermark generation, embedding and detection processes. Threshold cryptography used in the watermark generation process can enhance the robustness, since in order to recover the embedded watermark, only parts of watermark slices are required. Watermarks are embedded in a given layer-based web document through vertically moving the occupied layers, which guarantees watermark embedding cause no degradation on web display. The advantage of the detection process is that it can recover the lost watermark slice property by its next layer.

(6) Modify the capitalization of the tag name characters[6]. This scheme embeds watermarks through altering the case of letters in HTML tags (called upper-lower coding, or ULC). Obviously, this will not increase the file size of web pages. Specifically, the 0s and 1s in watermark $W_i, i = 1, 2, ..., R$, define the case of letters in the HTML tags in the ith text line T_i: the j th letter is set to be lower case if the j th element of W_i is '0', otherwise it is set to be upper case. When the number of letters is larger than the length of W_i, the index of the letter is performed with the mod operation by the length of W_i.

(7) Using some ending tags to hide information, or considering some tags can have two types of equivalent formula, and we can use different code formats to embed watermark information [7-9] similar to excel watermarking [10].Because a webpage presentation can be coded by different ways, using an HTML webpage to cover secret message can be done by adjusting tag attributes. In [7], an improved webpage data hiding method in terms of embedding capacity was proposed. With the advanced development of mobile devices and wireless networking, HTML announces a new version of webpage coding rule. In addition, cascading style sheets provide a lot of supports to help designers to create colorful webpages. Based on the flexibility of webpage coding, this method utilizes the property of different font size settings to hide secret data. Thus, the stego webpage is most similar to the original one even the webpage's source code has been modified based on the embedding rules. This method not only can achieve the goal of the secret data delivery but also improve the embedding capacity.

There are obvious deficiencies in the above embedding methods for web pages, most of them have the characteristics of insufficient watermark embedding capacity and poor robustness. The method of changing the case or changing the number of spaces will make the case or the number of spaces in the HTML code constantly change, which is easy to be recognized, the concealment of the watermark is not good, and the anti-detection ability is very weak. Through changing all letters to lowercase or changing the number of spaces to one, you can easily remove the watermark. For the method of using non-existing tags, because the tags in the HTML code are fixed, the non-existing tags can be easily identified, and the concealment and anti-attack capabilities are not strong. Although the last method has better concealment, it has few code embedding points and small watermark information capacity. It is still a direction of active development in the field of web watermarking technology.

The rest of this paper is organized as follows. Section 2 introduces some related works. The detailed description of our proposed methods is then presented in Section 3. In Section 4, the experimental results and analysis are reported. Finally, the conclusion is provided in Section 5.

2. **Related Works.**

Web page watermarking methods are mainly used for content authentication or copyright watermarking. Here, we introduce two typical web page watermarking schemes, one is for content authentication, and the other is for copyright protection.

2.1. **Web Page Watermarking for Authentication.**

Zhang et al.[3] proposed a fragile watermarking scheme based on hash function for web pages. They considered that there are two main features in an HTML file, one is that the superfluous 'Space' and 'Tab' characters are passed over by web browsers automatically; the other is that the tags are not case sensitive. Zhang et al.[3] generated two kind watermarks called word watermark and line watermark and then embedded them into HTML by utilizing the above-mentioned two features.

2.1.1. *Watermark Generating.*

In the process of generating watermarks, two components, i.e., word watermark and line watermark are produced as follows:

(1)Word watermark generation. Assume that there are U words in the HTML code. For each word, the SHA-1 algorithm is first performed on it to encrypt it using the same secret key Key_1. Thus, U 160-bit character strings can be obtained. Then, U integers s_i, $i = 1, 2, \ldots, U$ are generated by accumulating all the ASCII values of characters in each character string. After that, the U accumulated values are adopted as the seeds of the pseudo random function to get U '0-1' sequences. Considering the average length of Huffman Coding, we take the length of every '0-1' sequence as $\lfloor log_2 A \rfloor$, where A is the total number of letters in the alphabet. For example, for English alphabet, including upper and lower cases, $A = 52$. Therefore, each word watermark has a '0-1' sequence of length 6. The U '0-1' sequences can be expressed as $W_W(W_i) = Random(s_i, Key_1), i = 1, 2, \ldots, U$, where $W_W(W_i)$ is the word watermark of the $i-$th word W_i, *Random* is the random function to generate pseudo random sequences, s_i is the accumulated value generated from the $i-$th word, Key_1 is a private key. Furthermore, we transform these '0-1' sequences into some 'Space-Tab' sequences respectively and embed them into a web page utilizing the invisibility of browsers. The specific conversion method is 0 for 'Space' and 1 for 'Tab'. After performing this operation, we can obtain word watermarks to be embedded later.

(2)Line watermark generation. Assume that there are V lines in the HTML code. For each line, the SHA-1 algorithm is utilized to encrypt it based on the same secret key Key_2. Thus, V 160-bit character strings can be obtained. Then, V integers t_j, $j = 1, 2, \ldots, V$ by accumulating all the ASCII values of characters in each character string. After that, the V accumulated values are adopted as the seeds of the pseudo random function to obtain V '0-1' sequences. Here, like the length of word watermarks, the length of line watermarks is also set as 6, that is, each line watermark has a '0-1' sequence of length 6. The V line watermarks can be expressed as $L_W(L_j) = Random(t_j, Key_2)$, $i = 1, 2, \ldots, V$, where $L_W(L_j)$ is the line watermark of the $j-$th line L_j, *Random* is the random function to generate pseudo random sequences, t_j is the accumulated value generated from the $j-$th line, Key_2 is a private key. Furthermore, the above-mentioned '0-1' sequences will be embedded into the random tags and locations respectively since the tags are case-insensitive and the superfluous 'Space' and 'Tab' characters will be ignored by web browsers automatically.

2.1.2. *Watermark Embedding.*

In the watermark embedding stage, word watermarks and line watermarks are embedded respectively. The specific steps are as follows:

(1) Word watermark embedding. Assume that there are U words in a web page, that is, there are U locations of word watermark embedding. Here, the corresponding embedding

locations are denoted as $1, 2, \dots, M$. Then a sequence of new locations $i_1, i_2, \dots, i_M$ are generated by a pseudo random function controlled by a secret key Key_3. Their relationship is $1 \leftrightarrow i_1, 2 \leftrightarrow i_2, \dots, j \leftrightarrow i_j, \dots, U \leftrightarrow i_U$. The word watermark of the j-th word is embedded just after the i_j-th word, $j = 1, 2, \dots, U$.

(2) Line watermark embedding. Assume that there are V lines in a web page, and $1, 2, \dots, V$ are used to denote these V embedding locations for line watermarks. Then, a sequence of new location $i_1, i_2, \dots, i_N$ are generated by a pseudo random function controlled by a secret key Key_4. Their relationship is $1 \leftrightarrow i_1, 2 \leftrightarrow i_2, \dots, t \leftrightarrow i_t, \dots, V \leftrightarrow i_V$. The line watermark of t-th line is embedded into the tag and end of the i_t-th line, $t = 1, 2, \dots, V$. The specific steps are based on the relationship between the length of each line watermark Len_t and the number of tag characters Num_t in the corresponding line ($t = 1, 2, \dots, V$) as below: (i) If $Len_t > Num_t$, the excess watermark ($Len_t - Num_t$) bits will be embedded into the end of line by altering them into 'Space-Tab' sequence. (ii) If $Len_t < Num_t$, the line watermark is embedded into the tags by circular method. (ii) If $Len_t = Num_t$, all the line watermarks will be embedded into the tags.

2.1.3. *Watermark Extraction and Authentication.* To judge if a watermarked web page has been tampered or not, the word watermarks and line watermarks should be regenerated, denoted as $WW'_i (i = 1, 2, 3, \dots, U)$ and $LW'_j (j = 1, 2, 3, \dots, V)$ respectively. At the same time, the watermarks are extracted from the watermarked web page. The specific principles are as follows: (1) Bit '1' is extracted for an upper case letter and bit '0' for a lower case letter; (2) The 'Space's and 'Tab's behind the words or lines are extracted as '0' and '1' respectively.

In the extraction process, there are two kind watermarks, the specific extraction processes are as follows: (1) Word watermarks will be extracted from the watermarked web page according to the pseudo random function controlled by the secret key Key_3, denoted as $WW''_i (i = 1, 2, 3, ..., U)$. (2) Line watermarks will be extracted from the watermarked web page according to pseudo random function controlled by the secret key Key_4, denoted as $LW''_j (j = 1, 2, 3, ..., V)$. During the line watermark extraction, two circumstances should be dealt with as follows: (i) If $Len_t < 6$, the watermark will be extracted from the end of the line until the length is six. (ii) If $Len_t > 6$, the excess watermark bits will be removed.

After all the word watermarks and line watermarks are extracted, the authentication process can be performed. There are two possible cases as follows: (1) In the watermarked web page, the words are tampered but the watermarks are unchanged, then we have $WW'_i \neq WW''_i (i = 1, 2, 3, \dots, U)$ and $LW'_j \neq LW''_j (j = 1, 2, 3, \dots, V)$. (2) In the watermarked web page, the watermarks are tampered but the words are unchanged, then we have $WW'_i \neq WW''_i (i = 1, 2, 3, \dots, U)$ and $LW'_j = LW''_j (j = 1, 2, 3, \dots, V)$, or $WW_i = WW''_i (i = 1, 2, 3, \dots, U)$ and $LW'_j \neq LW''_j (j = 1, 2, 3, \dots, V)$.

2.2. **Web Page Watermarking for Copyright Protection.** Mir [1] proposed a method to embed an invisible watermark into a webpage using the structural features of HTML (Hyper Text Markup) language. The textual content of the input web page is first parsed to generate watermarks, according to a predefined semantic and syntactic rule. Then, the watermarks are encrypted by the HASH algorithm to generate a hash function. After that, the hash results are replaced using invisible control characters to generate the set of watermark to be embedded. In the algorithm, a specific list of verbs (is, are), articles (a, an), and frequently occurring initial letters (wh, th) of English language are considered only. HASH is used for encryption purposes and unicode controlled characters for invisibility of watermark. HTML $< meta >$ tag has been selected as a cover file source location to embed the invisible watermark. The embedding process consists of four basic steps, i.e., watermark generation, watermark hashing, hash conversion, embedding. These four main stages are explained below in steps:

Step 1. Watermark generation.

Step 1.1. Read a URL and scan the text. A URL is subjected to the system, which feeds the text data to the following steps.

Step 1.2. Generate a watermark based on semantic, syntactic or any rules. Predefined rules for a language are extracted from the webpage textual content that constructs the initial watermarks.

Step 2. Watermark hashing.

Step 2.1. Take the text data in form of string values.

Step 2.2. Process the text data into 512 bits consecutive chunk (TextCHK).

Step 2.3. Break this 512 chuck (TextCHK) into 16 chuncks of 32 bit words

Step 2.4. For TextCHK, initialize the HASH value.

Step 2.5. Get the TextCHK's HASH.

Step 2.6. Trim the TextCHK's HASH into 8 digits: (i) Take the variable-length MsgCHK HASH;(ii) Convert it into binary digits;(iii)Trim the values;(iv) Get the concatenated fixed length 8-digit HASH value.

Step 3. Converting 8-digit HASH into white space using controlled characters

Step 3.1. Take the 8-digit HASH value.

Step 3.2. Read the first digit using "u200A", convert the first digit into white space using "u202F" and end the process by using "u205F" controlled character.

Step 3.3. Repeat the process 8 times

Step 4. Watermark Embedding.

Step 4.1. Take the white space HASH-ed watermark.

Step 4.2. Embed it into the meta tag of the HTML source code.

The extraction and validation process for the watermark are expressed as follows. First it takes white spaces from the $< meta >$ tag of HTML source code and then reads the corresponding values before decrypting the watermarks. A hash code is recovered and match with the original for the validation process. Original and generated watermarks are then ready to be verified for the originality based on the level of similarity between these two watermarks.

3. **Proposed Algorithm.**

There are lots of web watermarking schemes in literatures, however, most of them face only one purpose, e.g., copyright protection or content authentication. For this situation, this paper proposes a Multipurpose Multilevel Multichannel Information Hiding (MMMIH) framework for Web pages so as to fulfil different purposes simultaneously. Based on the proposed framework, we can provide protection and tracing purposes based on four-level independent watermark embedding in different channels (i.e., different embedding domains) of a Web page. The example interface of MMMIH for Web pages is given in Fig. 1. The interface is composed of four parts, i.e., original Web module, original watermark module, watermarked Web module, extracted watermark module. The 'Browse' button is used for selecting files, while the 'Save as' button for saving files. The 'Information' edit control is used for showing the related information, e.g., the file path of the opened web page, or the NC values of extracted watermarks. The 'Original Web' control is used for showing the opened original Web page or possibly watermarked Web pages. The 'Original Watermark' controls display different watermarks, i.e., robust watermark, extra information and three level fingerprints. One advantage of our framework is that we can implement multilevel multichannel embedding, i.e., different level watermarks are independent since they are embedded by utilizing different properties (i.e., based on different channels). There are four modes for 'embed mode' or 'extract mode', i.e. {CP, FP1, FP2, FP3}. 'CP' means copyright watermark (may includes 'extra information'), 'FP1' means fingerprint level 1, 'FP2' means fingerprint level 2 and 'FP3' means fingerprint level 3 respectively. If we press the 'Embed' or 'Extract' button, a dialog will appear to determine the embedding or extraction parameters. The 'Watermarked Web' control is used for displaying the opened watermarked or suspect Web files.

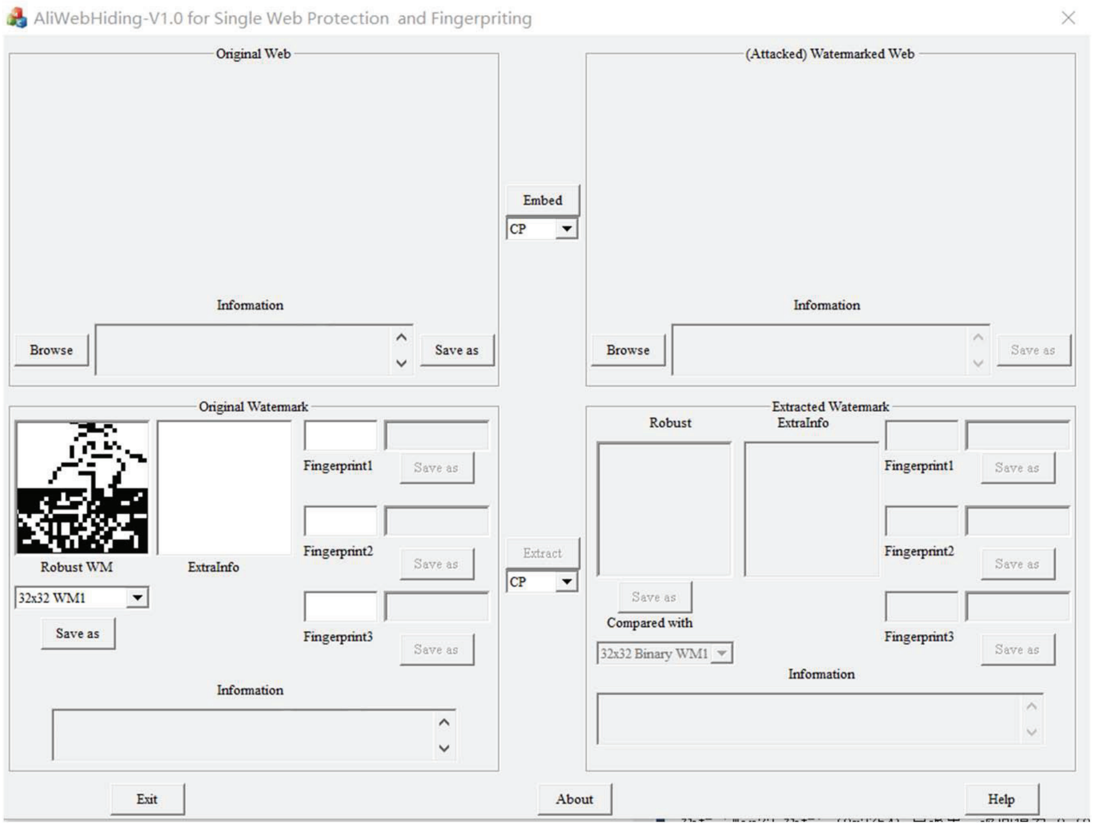


Figure 1. The MMMIH interface for WEB pages

Most of the methods studied for digital text watermarking rely on the content itself, not on the medium. Web page watermarking solves the problem of protecting not only the content but also the carrier medium itself by taking into account the structural elements of the page. In our framework, we design three kinds of watermarking schemes for Web pages, which are: (1)MMMIH based on the images in the web page; (2)MMMIH based on the scripts in the WEB page; (3)Web watermarking based on webpage's background with tiling. For the first and the second kinds of schemes, multipurpose independent watermarks can be embedded. For the last kind of scheme, only one level watermark can be embedded.

3.1. **MMMIH Based on Images in Web Pages.** When we build a website or optimize a website, we will add some pictures to the website. If it is just text information, most users will feel boring on the website, and they want to exit the website after reading the website. If you add pictures to the website, users will not have visual fatigue, which will cause customers continue browsing with interest. Here, we present a web watermarking method based on images as follows:

3.1.1. *Embedding Process.* The embedding process can be described as follows:

Step1: Copy the original html(htm) web page document to the "Results" directory, and change the file name to temp_ $**$.html(htm) according to the embedding mode (for CP mode, ** means CP; for FP1 mode, ** means FP1; for FP2 mode, ** means FP2; for FP3 mode, ** means FP3);

Step2: Copy the files associated with the original html (htm) to the "Results" directory, and modify the file name to temp_ $**$_files (** denotes the embedded mode, i.e., for CP, ** means CP; for FP1, ** means FP1; for FP2, ** means FP2; for FP3, ** means FP3);

Step3: Convert the temp_$**$.html(htm) document into a txt document, and modify the filename of the files corresponding to the original html(htm) to temp_$**$_files (for example, the original webpage files are samples_files, then the files after embedding the watermark should be modified to temp_ $**$_files), to ensure that the webpage after embedding the watermark can be associated with the css and js in the corresponding file folders, and after the modification, convert the txt document into html (htm) format;

Step4: Obtain data, take 0/1 of the information (image or image + data) to be embedded in sequence, and obtain a sequence of 0/1;

Step5: Generate a chaotic sequence according to the input password, and the original 0/1 sequence is scrambled in the order of the generated chaotic sequence;

Step6: Add the header identification bits (twenty 0s) to the front of the scrambled sequence;

Step7: The images are traversed over the temp_$**$_files folder in the "Results" directory , and all suitable image carriers are filtered out according to the size of the embedded information, the upper limit of the number of selected images is 40 (adjustable, macro definition);

Step8: If there is no suitable image carrier, then exit, otherwise continue;

Step9: The scrambled sequence is embedded in all suitable images based on dither modulation, and one or three times of embedding can be automatically selected according to the size of the carrier (redundant embedding to enhance the robustness);

Step 10: Complete the embedding.

3.1.2. *Extraction Process.* The extraction process can be described as follows:

Step1: Traverse the images in the temp_ $**$_files folder in the "Results" directory, and filter out all suitable image carriers according to the size of the embedded information;

Step2: Pre-read the image carrier to obtain the first 20 bits of embedded information, which are used to determine whether the image is embedded with watermark (if the number of '1's is less than 3, then the image is embedded with watermark). If the image satisfies the condition, extract all the information from the image;

Step3: Take the average value for each bit of all the extracted information (since the watermark may be embedded in multiple times), if the average value for a certain bit is greater than 0.5, the bit is considered to be 1, otherwise it is 0;

Step4: Get the 0/1 sequence, and inversely scramble it according to the chaotic sequence generated by the password to obtain the extracted 0/1 sequence;

Step5: Convert the extracted sequence 0/1 to the original information (image or image + data);

Step6: Complete the extraction.

3.1.3. *Dither Modulation Algorithm.* Dither modulation algorithm is a famous image watermarking scheme. Thus, we use it in our algorithm. The process can be illustrated as follows:

(1)Embedding process

Step1: After obtaining the jpg image, convert it to an RGB two-dimensional array;

Step2: Convert the RGB array to a YCbCr array, and select the Y channel for embedding;

Step3: Perform the DCT transformation on the Y channel matrix to obtain the coefficient matrix, divide the coefficient matrix into 8*8 blocks, and embed the information for each block;

Step4: Select a few bits in the block to embed according to the parameter selection. In this algorithm, each 8*8 block can embed a total of 8 bits information (the number of embedded bits is optional, but 8 bits are currently the best), where each 2 bits are assigned to different modes (0 for CP, 1 for FP1, 2 for FP2, 3 for FP3);

Step5: Inverse the coefficient matrix of the embedded content through DCT Transform to obtain the Y channel matrix, and then restore it to an RGB array through calculation, and then store it as a jpg file by the storage function, complete the embedding.

(2)Extraction process

Step1: After obtaining the jpg image, restore it to a two-dimensional RGB array;

Step2: Convert the RGB array to a YCbCr array, and select the Y channel for extraction;

Step3: Perform DCT transformation on the Y channel matrix to obtain a coefficient matrix,and the coefficient matrix is divided into 8*8 blocks, and information is extracted for each block;

Step4: Select a few bits in the block to extract according to the parameter selection, where each 2 bits are assigned to different modes (0 for CP, 1 for FP1, 2 for FP2, 3 for FP3), finish the extraction.

3.2. **Web Watermarking Based on Scripts in Web Pages.**

We design three methods to embed watermarks based on the scripts in the Web page. The first method is to embed watermark bits in web page comments. The second method is to embed watermark bits in “div display:none” of the web page html(htm). The third method is to embed the watermark image directly in the web page html(htm) <script>.

3.2.1. *Embedding Watermark Bits in Webpage’s Comments.*

In a web page, there is an element that is not displayed on the page, that is, the comment text of the code. Appropriate comments can help users better understand the division of each module in the web page, and also help to check and modify the code later. Annotating code is a good programming habit. In HTML5 documents, comments are divided into three categories: comments in the file start tag <html</html>, comments in CSS Cascading Style Sheets, and comments in JavaScirpt. There are two forms of comments in JavaScirpt. Here, we present a web watermarking based on comments as follows:

(1)Embedding process

Step1: Copy the original html(htm) web page document to the “Results” directory, and change the file name to temp_∗∗.html(htm) according to the embedded mode (for CP mode, ** means CP; for FP1 mode, ** indicates FP1; for FP2 mode, ** indicates FP2; for FP3 mode: ** indicates FP3);

Step2: Copy the files associated with the original html (htm) to the “Results” directory, and modify the file name to temp_∗∗_files (here, ** denotes the embedded mode, i.e., for CP, ** means CP; for FP1, ** means FP1; for FP2, ** means FP2; for FP3, ** means FP3);

Step3: Obtain data, take 0/1 of the information (image or image + data) to be embedded in sequence, and obtain a sequence of 0/1;

Step4: Generate a chaotic sequence according to the input password, and the original 0/1 sequence is scrambled in the order of the generated chaotic sequence;

Step5: Convert the scrambled 0/1 sequence to hexadecimal and stringify it;

Step6: Convert the temp_∗∗.html(htm) document into a txt document, and modify the filename of the files corresponding to the original html(htm) to temp_∗∗_files (for example, the original webpage files are samples_files, then the files after embedding the watermark should be modified to temp_∗∗_files), to ensure that the webpage after embedding the watermark can be associated with the css and js in the corresponding file folders. At the same time, select </body> in the txt document to embed <–CP000****–>. (CP000 means robust watermark; FP100 means Level 1 fingerprint; FP200 means Level 2 fingerprint; FP300 means Level 3 fingerprint watermark; **** represents a hexadecimal string). After modification, convert the txt file into html (htm) format;

Step7: Finish embedding.

(2)Extraction process

Step1: Open the html (htm) document, convert it into a txt document, traverse each line in the txt, search "CP000 (FP100, FP200, FP300)", and obtain the following hexadecimal string;

Step2: If it cannot be obtained, exit, indicating that there is no embedded information in the html (htm) document or the extraction code is incorrect;

Step3: Find "CP000 (FP100, FP200, FP300)" and extract the hidden information behind;

Step4: Convert the hidden information to hexadecimal numbers, and then convert it to a 0/1 sequence;

Step5: Get the 0/1 sequence, and inversely scramble it according to the chaotic sequence generated by the password to obtain the extracted 0/1 sequence;

Step6: Restore the extracted sequence 0/1 to the original information (image or image + data);

Step7: Complete the extraction and convert it back to html (htm) format.

3.2.2. *Embedding Watermark Bits in "Div Display:none".* In a Web page file, every element has a default display value, such as inline-block, block, table, etc. To hide an element with display property, we should use "display: none". When an element is hidden with "display: none", all its descendants are removed. We design a web watermarking method based on this property as follows:

(1) Embedding process

Step1: Copy the original html(htm) web page document to the "Results" directory, and change the file name to temp_∗∗.html(htm) according to the embedded mode (for CP mode, ** means CP; for FP1 mode, ** indicates FP1; for FP2 mode, ** indicates FP2; for FP3 mode, ** indicates FP3);

Step2: Copy the contents of the files folder associated with the original html (htm) to the Results directory, and modify the file name to temp_∗∗_files (here, ** denotes the embedded mode, i.e., for CP, ** means CP; for FP1, ** means FP1; for FP2, ** means FP2; for FP3, ** means FP3);

Step3: Obtain data, take 0/1 of the information (image or image + data) to be embedded in sequence, and obtain a sequence of 0/1;

Step4: Generate a chaotic sequence according to the input password, and the original 0/1 sequence is scrambled in the order of the generated chaotic sequence;

Step5: Convert the scrambled 0/1 sequence to hexadecimal and stringify it;

Step6: Convert the temp_∗∗.html(htm) document into a txt document, and modify the filename of the files corresponding to the original html(htm) to temp_∗∗_files (for example, the original webpage files are samples_files, then the files after embedding the watermark should be modified to temp_∗∗_files), to ensure that the webpage after embedding the watermark can be associated with the css and js in the corresponding file folders.

At the same time, randomly select a line in the txt document to embed <div style = “display:none”></div>CP000********<div>. (CP000 means robust watermark; FP100 means Level 1 fingerprint; FP200 means Level 2 fingerprint; FP300 means Level 3 fingerprint watermark; **** represents a hexadecimal string). After modification, converted to html (htm) format;

Step7: Finish embedding.

(2)Extraction process

Step1: Open the html (htm) document, convert it into a txt document, traverse each line in the txt, search “CP000 (FP100, FP200, FP300)”, and obtain the following hexadecimal string;

Step2: If it cannot be obtained, exit, indicating that there is no embedded information in the html (htm) document or the extraction code is incorrect;

Step3: Find “CP000 (FP100, FP200, FP300)” and extract the hidden information behind;

Step4: Convert the hidden information to hexadecimal numbers, and then convert it to a 0/1 sequence;

Step5: Get the 0/1 sequence, and inversely scramble it according to the chaotic sequence generated by the password to obtain the extracted 0/1 sequence;

Step6: Restore the extracted sequence 0/1 to the original information (image or image + data);

Step7: Complete the extraction and convert it back to html (htm) format.

3.2.3. *Embedding Watermark Images Directly in Webpage’s Html(htm) Script.* The <script> tag is used to define client-side scripts such as JavaScript. The script code can be placed directly in the <script> element, or an external script file can be referenced through the “src” attribute of <script>. The required attribute type is used to indicate the MIME type of the script. JavaScript is often used for manipulating images, validating forms, and dynamically modifying content. Here, we present a watermarking method based on <script>s as follows:

(1)Embedding process

Step1: Copy the original html(htm) web page document to the “Results” directory, and change the file name to temp_$**$.html(htm) according to the embedded mode (for CP mode,** means CP; for FP1 mode, ** indicates FP1; for FP2 mode, ** indicates FP2; for FP3 mode: ** indicates FP3);

Step2: Copy the files associated with the original html (htm) to the “Results” directory, and modify the file name to temp_$**$_files (here, ** denotes the embedded mode, i.e., for CP, ** means CP; for FP1, ** means FP1; for FP2, ** means FP2; for FP3, ** means FP3);

Step3: Copy the watermark image to the “Results” directory, which is associated with the temp_$**$_files of the watermark html webpage;

Step4: Convert the temp_$**$.html(htm) document into a txt document, and modify the filename of the files corresponding to the original html(htm) to temp_$**$_files (for example, the original webpage files are samples_files, then the files after embedding the watermark should be modified to temp_$**$_files), to ensure that the webpage after embedding the watermark can be associated with the css and js in the corresponding file folders. At the same time, in the txt document, find the last ¡/body¿ (because the image is embedded, it must be displayed during extraction, so the hidden code must be placed in the position where the image can be displayed.), add “< script> var CP = document.createElement(“img”); CP.alt=“CP000”; CP.setAttribute(“src”,“****”); CP.style.position=“fixed”; CP. style.display= “none”; CP.style.top = 0 + “px”; CP.style.left = 0 + “px”; CP.style.zIndex = 100000;

document.body.appendChild(CP); </script>". Here, **** means the path of the image. After modification, the txt document is converted back into html (htm);

Step5: Complete the embedding.

(2)Extraction process

Step1: Open the html (htm) document, convert it into a txt document, traverse each line in the txt, search "CP000 (FP100, FP200, FP300)", and obtain the following hexadecimal string;

Step2: If it cannot be obtained, exit, indicating that there is no embedded information in the html (htm) document or the extraction code is incorrect;

Step3: Find "CP.style.position = "fixed";CP.style.display = "none"";

Step4: Replace "CP.style.position= "fixed";CP.style.display = "none";" with "CP.style.position= "fixed";"

Step5: Convert back to html (htm) format.

Step6: Open the extracted html (htm) document, and you can see the original watermark image.

3.3. **Web Watermarking Based on Webpage's Background with Tiling.** We design several methods based on webpage's backgrounds. The first method is to embed strings with tiling using the canvas brush. The second method is to embed barcodes horizontally and tile with horizontal lines. The third method is to embed barcodes vertically and tile with vertical lines.

3.3.1. *Embedding Strings with Tiling Using the Canvas Brush.* New to HTML5, <canvas> is an HTML element in which images can be drawn using script (usually JavaScript). It can be used to create photo collections or simple (and not so simple) animations, or even real-time video processing and rendering. Here, we design a method to hiding strings in the canvas as follows:

(1)Embedding process

Step1: Copy the original html(htm) web page document to the "Results" directory, and change the file name to temp_ ∗ ∗.html(htm) according to the embedded mode (for CP mode,** means CP; for FP1 mode, ** indicates FP1; for FP2 mode, ** indicates FP2; for FP3 mode: ** indicates FP3);

Step2: Copy the files associated with the original html (htm) to the "Results" directory, and modify the file name to temp_ ∗ ∗_files (here, ** denotes the embedded mode, i.e., for CP, ** means CP; for FP1, ** means FP1; for FP2, ** means FP2; for FP3, ** means FP3);

Step3: Convert the temp_∗∗.html(htm) document into a txt document, and modify the filename of the files corresponding to the original html(htm) to temp_∗∗_files (for example, the original webpage files are samples_files, then the files after embedding the watermark should be modified to temp_ ∗ ∗_files), to ensure that the webpage after embedding the watermark can be associated with the css and js in the corresponding file folders. At the same time, find </body> in the txt document, and take the next line to embed as follows:

"<script> var dfColor = window.getComputedStyle(document.body, null).backgroundColor; var colorG = dfColor.slice(dfColor.indexOf('a') > -1 ? 5 : 4).slice(0, -1).split(', '); var d = Parameter1; var canvas = document.createElement('canvas'); canvas.width = 100; canvas.height = 100; var ctx = this.canvas.getContext('2d'); ctx.font = "18px Courier New"; console.log(colorG); if (colorG[3] == '0') ctx.fillStyle = "rgb(" + (255 - d) + ", " + (255 - d) + ", " + (255 - d) + ")"; else var color = "rgb" + (dfColor.indexOf('a') > -1 ? 'a' : ")" + "(" + (colorG[0] > d ? colorG[0] - d : 0) + ", " + (colorG[1] > d ? colorG[1] - d : 0) + ", " + (colorG[2] > d ? colorG[2] - d : 0) + (, colorG[3] ? ")" :

(", " + colorG[3] + ")")); ctx.fillStyle = color; console.log(ctx.fillStyle); ctx.rotate(-10 * Math.PI / 180); ctx.fillText("String", 0, 30); document.body.style.backgroundImage = 'url(' " + ctx.canvas.toDataURL() + " ')'; </script>".

Note that d is determined according to Parameter1 (its value is the chromaticity difference from the background), and "string" is the input string information;

Step4: Complete the embedding.

(2) Extraction process

Step1: Use the canny algorithm to process the screenshot, and intuitively see the embedded string information;

Step2: Complete the extraction.

3.3.2. *Embedding Barcodes Horizontally and Tiling with Horizontal Lines.* The method to embed barcodes horizontally and tile with horizontal lines can be described as follows:

(1)Embedding process

Step1: Copy the original html(htm) web page document to the "Results" directory, and change the file name to temp_ ∗ ∗.html(htm) according to the embedded mode (for CP mode,** means CP; for FP1 mode, ** indicates FP1; for FP2 mode, ** indicates FP2; for FP3 mode: ** indicates FP3);

Step2: Copy the files associated with the original html (htm) to the "Results" directory, and modify the file name to temp_ ∗ ∗_files (here, ** denotes the embedded mode, i.e., for CP, ** means CP; for FP1, ** means FP1; for FP2, ** means FP2; for FP3, ** means FP3);

Step3: Convert the temp_∗∗.html(htm) document into a txt document, and modify the filename of the files corresponding to the original html(htm) to temp_∗∗_files (for example, the original webpage files are samples_files, then the files after embedding the watermark should be modified to temp_ ∗ ∗_files), to ensure that the webpage after embedding the watermark can be associated with the css and js in the corresponding file folders. At the same time, find </body> in the txt document, and take the next line to embed as follows:

"<script>
document.body.style.backgroundImage="url('watermark_path');
document.body.style.backgroundRepeat="repeat-x";
document.body.style.backgroundPositionY = position_s+"px";
</script>".

Note that "watermark_path" is the address path of the barcode, "positon_s" is the distance between the embedded barcode and the top of the web page, which is determined by Parameter4. After modification, convert the txt document to html (htm);

Step4: The input string generates a barcode image according to Parameter1 (barcode height) and Parameter2 (barcode unit line width), and according to the input of Parameter3, replace the pixel value in the barcode with the value of Parameter3*3. Finally, an image that is lighter than the original barcode is obtained;

Step5: Put the barcode obtained by the final processing into the file corresponding to html;

Step6: Complete the embedding.

(2)Extraction process

Step1: Open the html (htm) document and locate the corresponding file;

Step2: Go through the pictures under the file and find the barcode picture (according to the picture name);

Step3: Replace the pixel value in the image equal to Parameter3*3 with 0;

Step4: Heighten the barcode image;

Step5: Scan the heightened barcode through the mobile phone software to obtain the watermark string information;

Step6: Complete the extraction.

3.3.3. *Embedding Barcodes Vertically and Tiling with Vertical Lines.* The method to embed barcodes vertically and tile with vertical lines can be described as follows:

(1)Embedding process

Step1: Copy the original html(htm) web page document to the "Results" directory, and change the file name to temp_ ∗ ∗.html(htm) according to the embedded mode (for CP mode,** means CP; for FP1 mode, ** indicates FP1; for FP2 mode, ** indicates FP2; for FP3 mode: ** indicates FP3);

Step2: Copy the files associated with the original html (htm) to the "Results" directory, and modify the file name to temp_ ∗ ∗_files (here, ** denotes the embedded mode, i.e., for CP, ** means CP; for FP1, ** means FP1; for FP2, ** means FP2; for FP3, ** means FP3);

Step3: Convert the temp_∗∗.html(htm) document into a txt document, and modify the filename of the files corresponding to the original html(htm) to temp_∗∗_files (for example, the original webpage files are samples_files, then the files after embedding the watermark should be modified to temp_ ∗ ∗_files), to ensure that the webpage after embedding the watermark can be associated with the css and js in the corresponding file folders. At the same time, find </body> in the txt document, and take the next line to embed as follows

"<script>
document.body.style.backgroundImage="url('watermark_path');
document.body.style.backgroundRepeat="repeat-y";
document.body.style.backgroundPositionY = position_s+"px";
</script>".

Note that "watermark_path" is the address path of barcode, "positon_s" is the distance between the embedded barcode and the left of the webpage, which is determined by Parameter4. After modification, convert the txt document to html (htm);

Step4: Based on the input string, we generates a barcode image according to Parameter1 (barcode height), Parameter2 (barcode unit line width), and according to the input of Parameter3, replace the pixel value in the barcode with the value of Parameter3*3. Finally, an image that is lighter than the original barcode is obtained;

Step5: Perform image transposition operations to obtain vertical bar codes;

Step6: Put the barcode obtained by the final processing into the file corresponding to html;

Step7: Embedding is completed.

(2)Extraction process

Step1: Open the html (htm) document and locate the corresponding file;

Step2: Go through the pictures under the file and find the barcode picture (according to the picture name);

Step3: Replace the pixel value in the image equal to Parameter3*3 with 0;

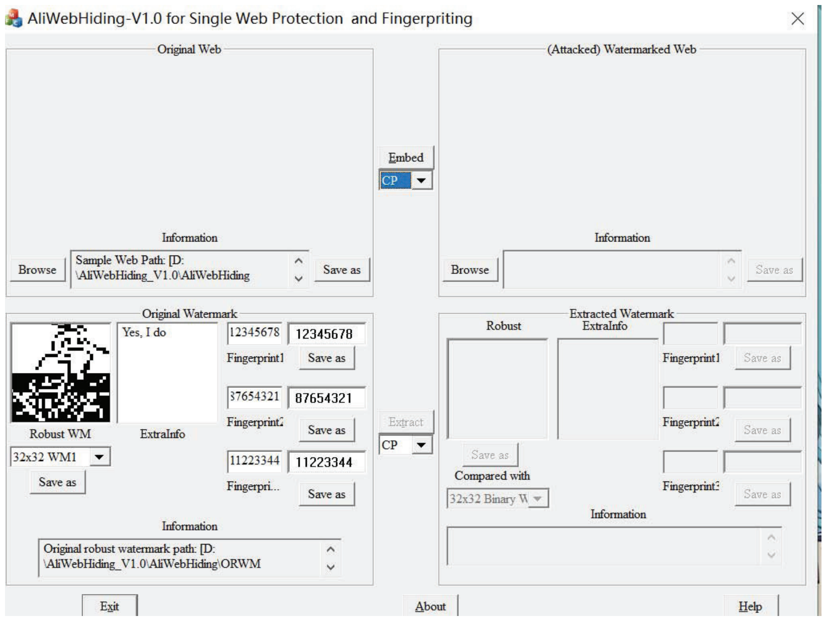
Step4: Transpose the barcode;

Step5: Heighten the barcode image;

Step6: Scan the heightened barcode through the mobile phone software to obtain the watermark string information;

Step7: Complete the extraction.

4. **Experimental Results.** In this section, we test the effectiveness of our MMMIH framework for web pages, where we embed different watermarks independently in different properties of the web page, i.e., multichannel embedding mode. In this framework, we totally presents seven methods. In the following experiments, we use the same watermark setting as shown in Fig. 2, where we at most embed four watermarks as given in Fig.3 in a web page, where CP (the robust watermark) includes the watermark image (bird) and extra information(“Yes, I do”), FP1 (Level 1 fingerprint) is the text image “12345678”,FP2 (Level 2 fingerprint) is the text image “87654321” and FP3 (Level 3 fingerprint) is the text image “11223344”. After testing the performance for seven methods respectively, we list the overall performance in Table 1, where Method 1 stands for “MMMIH Based on Images”, Method 2 stands for “Embedding in Webpage’s Comments”, Method 3 stands for “Embedding in “div display:none””, Method 4 stands for “Embedding Directly in Html(htm) Script”, Method 5 stands for “Embedding Strings Using the Canvas Brush”, Method 6 stands for “Embedding Barcodes Horizontally”, and Method 7 stands for “Embedding Barcodes Vertically”. In addition, “Edit Attack” means “Modify, delete, add statements or image files in the html script file”. From Table 1, we can see that all of the proposed seven methods can resist “Save As” attack, and they can resist some modifications in Web pages, and even some of them can resist screenshot and screen-cam attacks. In order to understand the performance of the algorithm in detail, the detailed experiments and performance analysis of three typical algorithms among the seven algorithms are given below.



Sample Web Path: [D:\AliWebHiding_V1.0\AliWebHiding

Embed: CP

Yes, I do

12345678 | 12345678

37654321 | 87654321

11223344 | 11223344

Robust WM: 32x32 WM1

Original robust watermark path: [D:\AliWebHiding_V1.0\AliWebHiding\ORWM

Extract: CP

Compared with: 32x32 Binary W

FIGURE 2. The watermark settings of following experiments

4.1. **Performance of the Proposed MMMIH Method Based on Images in Web Pages.** For any web page with many images, our first method can embed four level watermarks independently in images displaying in the web page. The example original web page and the four-level watermarked web page are shown in Fig.4. The average PSNR value between the original pictures in the original web page and the watermarked pictures in the watermarked web page is 36.12dB (after one level embedding, 40.35dB; after two level embedding, 38.41dB; after three level embedding, 37.24dB). From Fig. 4, we can see that the invisibility of Method 1 is very good. The extracted watermarks from

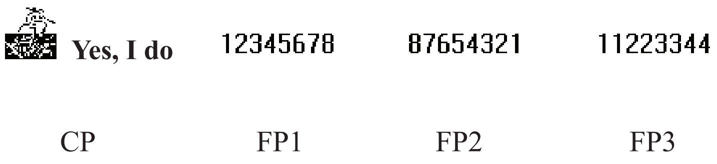
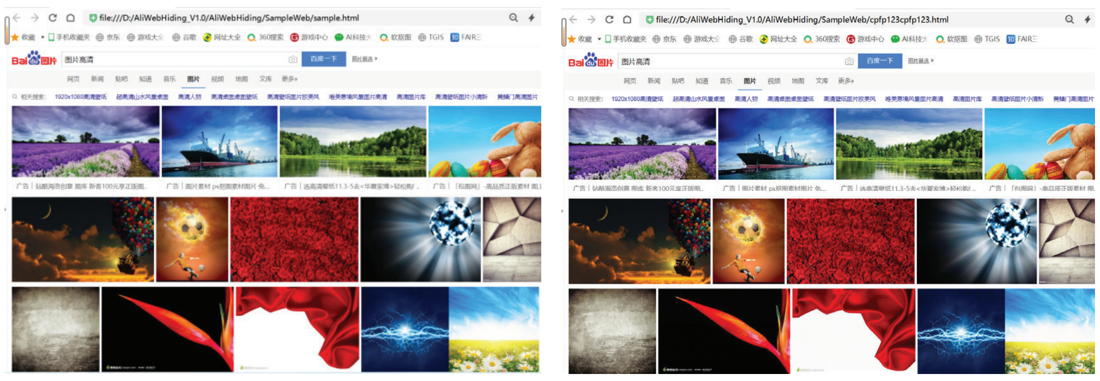


Figure 3. The four level watermarks to be embedded

Table 1. Test Results of Seven Web Information Hiding Methods

Method	Edit Attack	"Save As"	Screenshot	Screen-Cam
Method 1	Only all pictures removed, watermark lost	✓	X	X
Method 2	Only related comments removed, watermark lost	✓	X	X
Method 3	Only related "div" removed, watermark lost	✓	X	X
Method 4	Only related script removed, watermark lost	✓	X	X
Method 5	Only related canvas removed, watermark lost	✓	✓	X
Method 6	Only barcode removed, watermark lost	✓	✓	✓
Method 7	Only barcode removed, watermark lost	✓	✓	✓

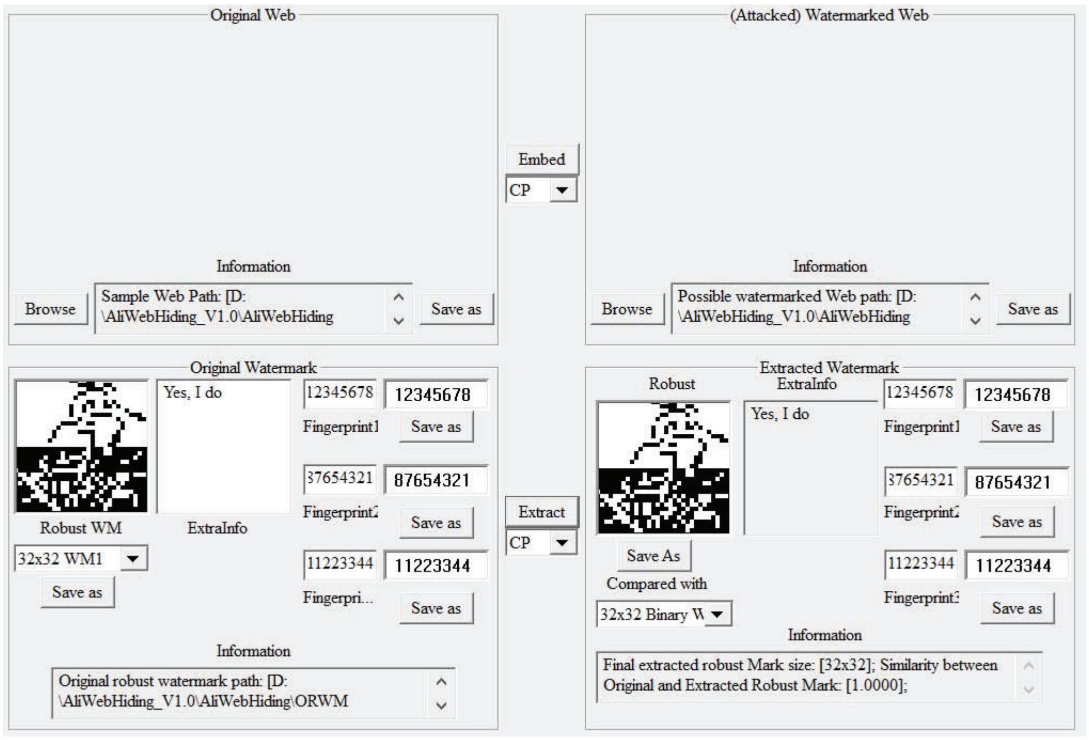
the watermarked web page is shown in Fig. 5. In this paper, we adopt NC to denote the similarity between two marks of the same size, where NC= 1 means two watermarks are identical, while NC= 0 means they are totally different. From Fig. 5, we can see that the four level watermarks (including the extra information "Yes, I do") can be completely extracted without any error from the watermarked web page without attack, i.e., NC=1.0 for each level watermark.


file:///D:/AliWebHiding_V1.0/AliWebHiding/SampleWeb/sample.html
file:///D:/AliWebHiding_V1.0/AliWebHiding/SampleWeb/cpfp123cpfp123.html

(a) Original web page (b) Four level watermarked web page

Figure 4. The original and four level watermarked web pages

In fact, we can extract each level watermark independently even if the pictures in the watermarked web page have undergone some image processing operations. Table 2 gives the extraction results in NC values between the extracted watermark and the original watermark of each level after all of the watermarked pictures in the watermarked web page has undergone several common image processing operations, including JPEG compression with QF=90, JPEG compression with QF=70, cropping the upper-left corner, scaling with factor 1.2, scaling with factor 0.8, adding Gaussian noises by 1% and median filtering with size 3×3. From Table 2, we can see that our scheme is robust to common image processing operations, and for every level, the NC value is above 0.92. From these results, we can see



Sample Web Path: [D: \AliWebHiding_V1.0\AliWebHiding

Possible watermarked Web path: [D: \AliWebHiding_V1.0\AliWebHiding

Yes, I do

12345678 12345678

37654321 87654321

11223344 11223344

Original robust watermark path: [D: \AliWebHiding_V1.0\AliWebHiding\ORWM

Final extracted robust Mark size: [32x32]; Similarity between Original and Extracted Robust Mark: [1.0000];

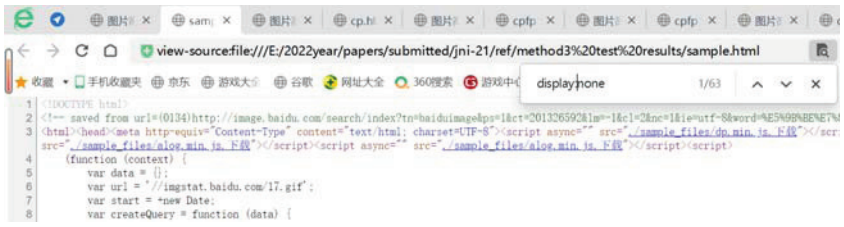
FIGURE 5. The extracted four level watermarks from the watermarked web page

that Method 1 is robust to most common image processing operations on the watermarked web page.

TABLE 2. NC values of four level watermarks extracted from the watermarked web page under different attacks.

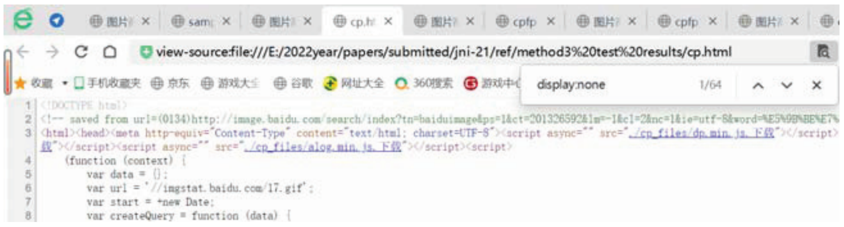
Level	CP	FP1	FP2	FP3
JPEG90	1.000	1.000	1.000	1.000
JPEG70	0.996	0.990	0.988	0.981
Cropping25%	0.975	0.971	0.967	0.965
Scaling 1.2	0.952	0.947	0.944	0.941
Scaling 0.8	0.961	0.958	0.955	0.950
Guassian noise 1%	0.964	0.961	0.958	0.954
Median filtering by 3×3	0.934	0.929	0.925	0.921

4.2. **Performance of Embedding Watermark Bits in "Div Display:none".** For any web page, we can embed four level watermarks independently in "div display:none". We still use the web page shown in Fig. 4(a) as our example. Fig. 6 shows the source code of the original web page and the source code of the watermarked web pages after different levels. If the four level watermarked web page is not attacked, then the extraction result is the same as shown in Fig. 5. If we remove some part of the source code of the four level watermarked web page randomly, we can also extract four watermarks without distortion when the code related to "display:none"s of embedded watermarks is not removed. Fig. 7 shows an example of the corresponding extraction result after 10% of the source code of the four level watermarked web page is removed. From Fig.7, we can see that the FP2 cannot be extracted since the source code related to FP2 is removed.


view-source:file:///E:/2022year/papers/submitted/jni-21/ref/method3%20test%20results/sample.html
display:none
1/63

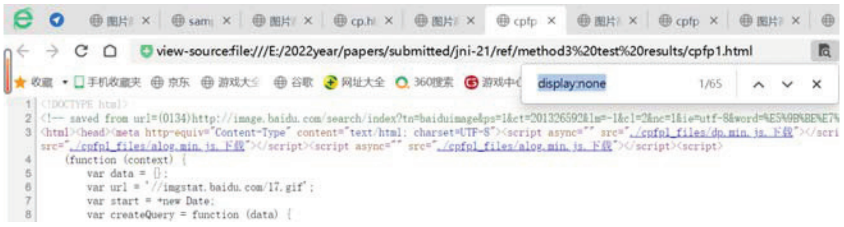
```
<!DOCTYPE html>
<!-- saved from url=(0134)http://image.baidu.com/search/index?tn=baiduimage&ps=1&ct=201326592&lm=-1&cl=2&nc=1&ie=utf-8&word=%E5%9B%BE%E7%
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"><script async="" src="./sample_files/dp.min.js.下载"></scr
src="./sample_files/alog.min.js.下载"></script><script async="" src="./sample_files/alog.min.js.下载"></script><script>
    (function (context) {
        var data = {};
        var url = '//imgstat.baidu.com/17.gif';
        var start = +new Date;
        var createQuery = function (data) {
```

(a) Source code of the original web page with 63 "display:none"s


view-source:file:///E:/2022year/papers/submitted/jni-21/ref/method3%20test%20results/cp.html
display:none
1/64

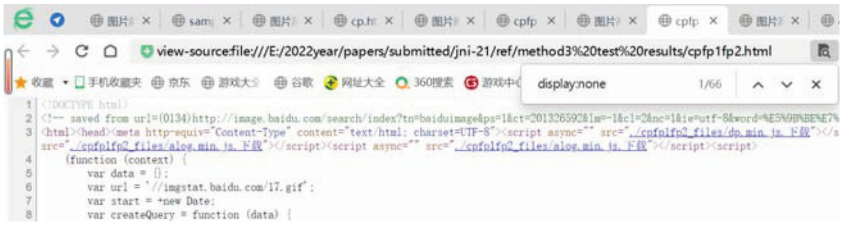
```
<!DOCTYPE html>
<!-- saved from url=(0134)http://image.baidu.com/search/index?tn=baiduimage&ps=1&ct=201326592&lm=-1&cl=2&nc=1&ie=utf-8&word=%E5%9B%BE%E7%
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"><script async="" src="./cp_files/dp.min.js.下载"></script>
载"></script><script async="" src="./cp_files/alog.min.js.下载"></script><script>
    (function (context) {
        var data = {};
        var url = '//imgstat.baidu.com/17.gif';
        var start = +new Date;
        var createQuery = function (data) {
```

(b) Source code of one level watermarked web page with 64 "display:none"s


view-source:file:///E:/2022year/papers/submitted/jni-21/ref/method3%20test%20results/cpfp1.html
display:none
1/65

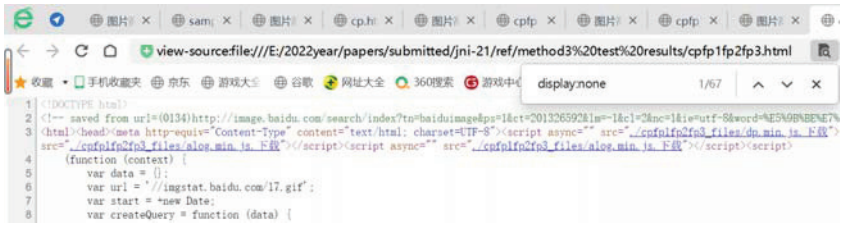
```
<!DOCTYPE html>
<!-- saved from url=(0134)http://image.baidu.com/search/index?tn=baiduimage&ps=1&ct=201326592&lm=-1&cl=2&nc=1&ie=utf-8&word=%E5%9B%BE%E7%
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"><script async="" src="./cpfp1_files/dp.min.js.下载"></scri
src="./cpfp1_files/alog.min.js.下载"></script><script async="" src="./cpfp1_files/alog.min.js.下载"></script><script>
    (function (context) {
        var data = {};
        var url = '//imgstat.baidu.com/17.gif';
        var start = +new Date;
        var createQuery = function (data) {
```

(c) Source code of two level watermarked web page with 65 "display:none"s


view-source:file:///E:/2022year/papers/submitted/jni-21/ref/method3%20test%20results/cpfp1fp2.html
display:none
1/66

```
<!DOCTYPE html>
<!-- saved from url=(0134)http://image.baidu.com/search/index?tn=baiduimage&ps=1&ct=201326592&lm=-1&cl=2&nc=1&ie=utf-8&word=%E5%9B%BE%E7%
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"><script async="" src="./cpfp1fp2_files/dp.min.js.下载"></s
src="./cpfp1fp2_files/alog.min.js.下载"></script><script async="" src="./cpfp1fp2_files/alog.min.js.下载"></script><script>
    (function (context) {
        var data = {};
        var url = '//imgstat.baidu.com/17.gif';
        var start = +new Date;
        var createQuery = function (data) {
```

(d) Source code of three level watermarked web page with 66 "display:none"s


view-source:file:///E:/2022year/papers/submitted/jni-21/ref/method3%20test%20results/cpfp1fp2fp3.html
display:none
1/67

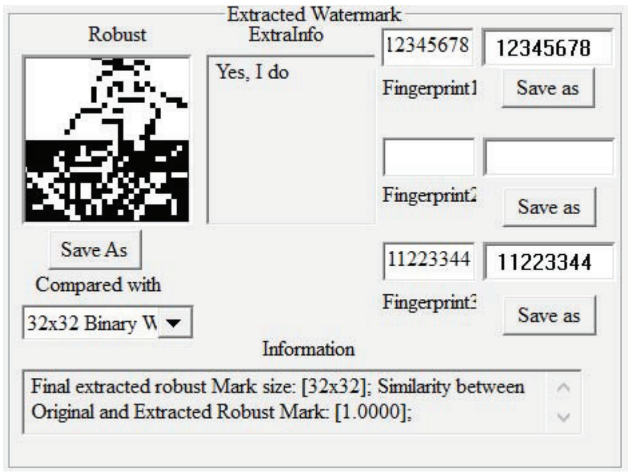
```
<!DOCTYPE html>
<!-- saved from url=(0134)http://image.baidu.com/search/index?tn=baiduimage&ps=1&ct=201326592&lm=-1&cl=2&nc=1&ie=utf-8&word=%E5%9B%BE%E7%
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"><script async="" src="./cpfp1fp2fp3_files/dp.min.js.下载">
src="./cpfp1fp2fp3_files/alog.min.js.下载"></script><script async="" src="./cpfp1fp2fp3_files/alog.min.js.下载"></script><script>
    (function (context) {
        var data = {};
        var url = '//imgstat.baidu.com/17.gif';
        var start = +new Date;
        var createQuery = function (data) {
```

(e) Source code of four level watermarked web page with 67 "display:none"s

FIGURE 6. The source codes of the original and watermarked web pages

4.3. Performance of Embedding Barcodes Horizontally and Tiling with Horizontal Lines.

In a web page, sometimes lines are designed to divided the web page into several sections. In this case, we can design barcodes according to the information bits to be embedded, and then tile this barcode to generate many barcodes and insert them as lines in the webpage. These barcodes are thin enough to make sure that we cannot notice the information in it. Only when we enlarger the barcode can we see the difference



ExtraInfo: Yes, I do

Fingerprint1: 12345678 | 12345678

Fingerprint2:

Fingerprint3: 11223344 | 11223344

Compared with: 32x32 Binary W

Information: Final extracted robust Mark size: [32x32]; Similarity between Original and Extracted Robust Mark: [1.0000];

FIGURE 7. The extraction results after source code removal by 10%

TABLE 3. Comparisons of our web page watermarking scheme with other two schemes

Method	Invisible	Anti-Copy	Anti-Screenshot	Anti-Screencam
[1]	yes	yes	no	no
[9]	yes	yes	no	no
our	yes	yes	yes	yes

between normal lines and barcodes. Fig. 8 shows the bar code generated from the information to be embedded "Alibaba", and this bar code is embedded as a dash line in the web page as shown in Fig. 9. Thus based on the bar code, even the web page is made a screenshot or made a screen-cam, we can also recognize the bar code. For example, we make a screen-cam on the watermarked web page and get the picture as shown in Fig. 10. We crop the dashed line from this picture and get the image as shown in Fig. 11, and we can just read the information from this dashed line easily. Table 3 compares this scheme with two existing schemes. From this table, we can see that our scheme performs better than existing schemes.

FIGURE 8. The generated barcode from “Alibaba”

5. **Conclusions.** This paper proposes a multipurpose multilevel multichannel framework for web page watermarking. Based on this framework, we develop seven watermarking schemes that can achieve multiple purposes including copyright protection, traitor tracing, document security management and so on. From experimental results, we can see that our proposed schemes can embed multiple watermarks and can extract all watermarks correctly and independently without attacks or even with some attacks. Future work will concentrate on further improving the performance and security by combining the watermarking scheme with other techniques [11-15].

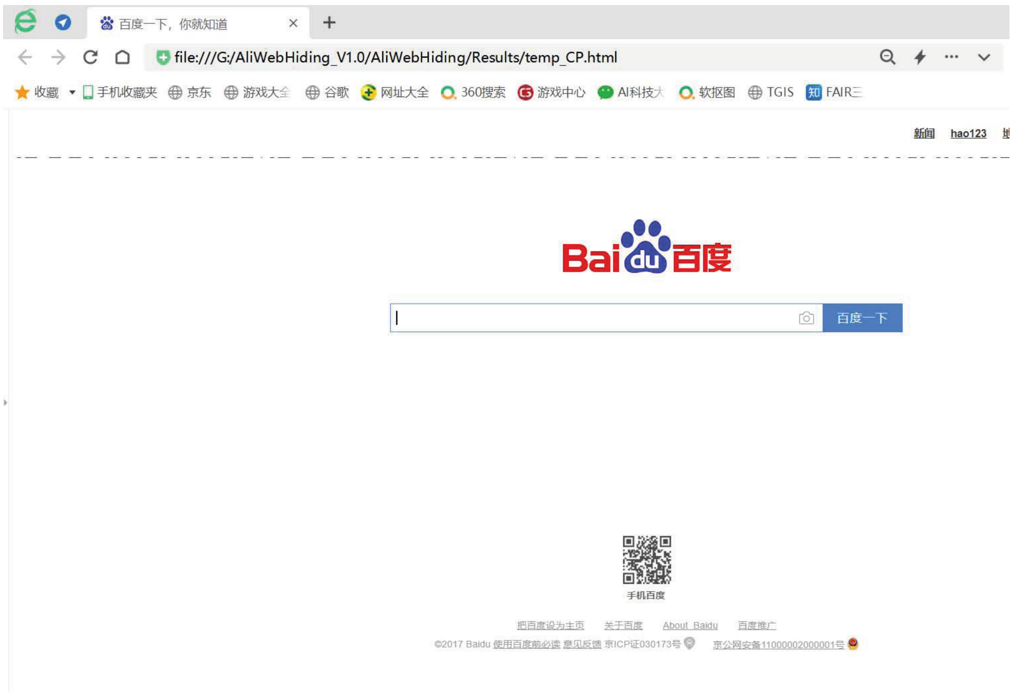

百度一下，你就知道
file:///G:/AliWebHiding_V1.0/AliWebHiding/Results/temp_CP.html
Baidu 百度
百度一下
手机百度

FIGURE 9. The watermarked web page with a tiled barcode line

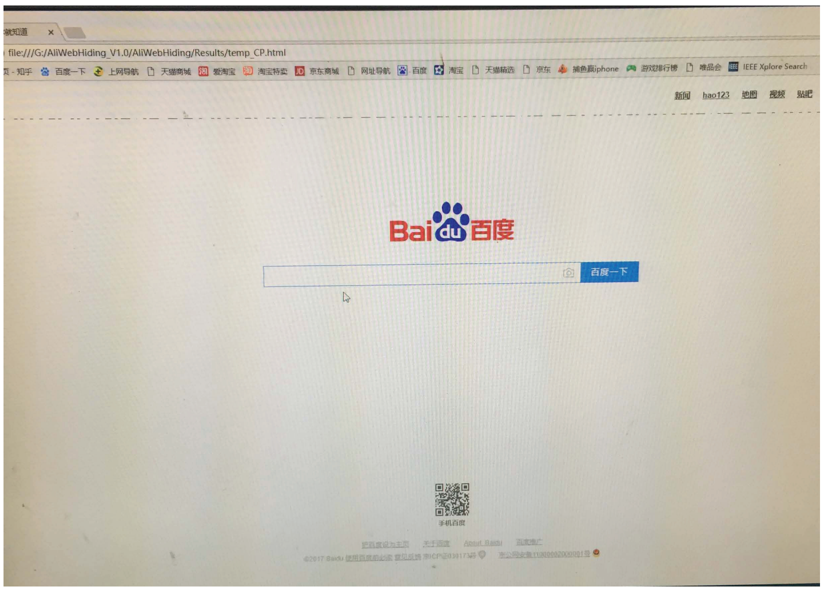


FIGURE 10. The screen-camed watermarked web page

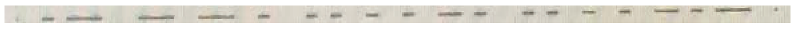


FIGURE 11. The cropped dashed line

Acknowledgment. This research is supported in part by the National Key Research and Development Program of China under Grant No.2020AAA0140004 and the Public Good Research Project of Science and Technology Program of Zhejiang Province under Grant No. LGG21F020005. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] N. Mir, “Copyright for web content using invisible text watermarking,” *Computers in Human Behavior*, vol.30, pp. 648–653, 2014.

[2] P. Sun, and H. T. Lu, “An efficient web page watermarking scheme,” *The second IEEE International Conference on Computer Science and Information Technology*, Beijing, pp. 163–167, 2009.

[3] Z. Zhang, H. Peng, and X. Long, “A fragile watermarking scheme based on hash function for web pages,” *International Conference on Network Computing and Information Security*, pp. 417–420, 2011.

[4] S. H. Low, N. F. Maxemchuk, and A. M. Lapone, “Document identification for copyright protection using centroid detection,” *IEEE Transactions on Communications*, vol. 46, no. 3, pp.372–383, 1998.

[5] D. Li, and B. Zhang, “DWTC: A dual watermarking scheme based on threshold cryptography for web document,” *International Conference on Computer Application and System Modeling*, pp. V8-510-V8-514, 2010.

[6] Q. Zhao, and H. Lu, “A PCA-based watermarking scheme for tamper-proof of web pages,” *Pattern Recognition*, vol. 38, no. 8, pp. 1321–1323, 2005.

[7] H. Liao, “A webpage data hiding method by using Tag and CSS attribute setting,” *Proceedings of the Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 122–125, 2014.

[8] S. Saini, “A survey on watermarking web contents for protecting copyright,” *International Conference on Innovations in Information, Embedded and Communication Systems (ICIBIECS) Coimbatore*, pp.1–4, 2015.

[9] R. J. Jaiswal, and N. N. Patil, “Implementation of a new technique for web document protection using unicode,”*In Proc. of the IEEE International Conference on Information Communication and Embedded Systems (ICICES)*, Chennai, India, pp. 69–72, 2013.

[10] D. Tian, Z.-M. Lu, and H.-Y. Fan, “A text watermarking algorithm based on hidden object,” *Journal of Information Hiding and Multimedia Signal Processing*, vol. 10, no. 3, pp. 470–478, 2019.

[11] T.-Y. Wu, X. Guo, Y.-C. Chen, S. Kumari, C.-M. Chen, “SGXAP: SGX-based authentication protocol in IoV-enabled fog computing,”*Symmetry*, vol. 14, no. 7, paper number 1393, 2022.

[12] T.-Y. Wu, Q. Meng, S. Kumari, and P. Zhang, “Rotating behind security: a lightweight authentication protocol based on IoT-enabled cloud computing environments”, *Sensors*, vol. 22, no. 10, paper number 3858, 2022.

[13] T.-Y. Wu, Q. Meng, L. Yang, X. Guo, and S. Kumari, “A provably secure lightweight authentication protocol in mobile edge computing environments,” *The Journal of Supercomputing*, vol. 78, pp. 13893–13914, 2022.

[14] T.-Y. Wu, Y.-Q. Lee, C.-M. Chen, Y. Tian, and N. A. Al-Nabhan, “An enhanced pairing-based authentication scheme for smart grid communications,” *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–13, 2021.

[15] C.-M. Chen, C.-T. Li, S. Liu, T.-Y. Wu, and J.-S. Pan, “A provable secure private data delegation scheme for mountaineering events in emergency system,” *IEEE Access*, vol.5, pp. 3410–3422, 2017.