Novi Sad J. Math.
Vol. 32, No. 1, 2002, 167-178

A NEW METHOD FOR WEB SITE PROTECTION AND COMPRESSION

Ivan Petković[1]

Abstract. Protection of the Web contents is of great importance in the field of web design. In this paper, a program called Web Encrypt 2, to provide protection and compression of web sites is presented. It is based on a modification of the trigraph algorithm and has several improvements in reference to the already existing software. Detailed description of the features and a comparative analysis of the proposed program are given.

AMS Mathematics Subject Classification (2000): 68P25

*Key words and phrases:*Internet, protection of Web sites, compression, encrypting, decoding.

1. Introduction

Today, the Internet has become a part of everyday life, including entertainment, science, information exchange, advertisement, e-commerce, etc. It is a common opinion that the Internet has created a new dimension in almost all life activities, providing easy and fast access to information that leads to the increase of general human knowledge. Internet promotion of the companies, agencies, banks, or some other institutions, as a new advertising strategy, has influenced enormous increase of Web sites (see, e.g., [1], [6]). For this reason, the amount of information available to the public has exponentially increased. As a direct consequence, the rate of information misuse has also increased.

Furthermore, every Web designer has its own methods of creating Web pages, using previously written scripts (e.g., client side JavaScript). His skills in writing the scripts and HTML code give him more freedom while designing a Web site (see `www.webdeveloper.com`), allowing him to create more efficient and visually better interface, and generally a higher quality Web site. In this way, Web site navigation will be easier and more efficient, increasing information value. Also, a well designed Web page does not have redundant code, which decreases download time. Taking advantages of skillful programming, many attractive visual effects can be created, (see, for example, `www.javascriptsource.com`). For all of the mentioned reasons, it is of importance to preserve originality, which leads to a need for the protection of code and scripts from the unlicensed using.

There are many aspects of Internet protection, see e.g., [2], [5], [7], [8]. The methods that are most frequently pursued today are directed to the protection of the data transfers from one location to another (e.g., online payment), see

[1]Faculty of Electronic Engineering, University of Niš, Beogradska 14, 18000, Niš, Yugoslavia, e-mail: ivan@design.co.yu

[2], [3], [4], [9]. Another approach to protection is focused to the client-side protection (text, images, HTML code and scripts). This procedure is very popular in the area of web design.

In this paper we present the software named Web Encrypt 2 which provides both the methods of protection and compression. The motivation for developing the program of this kind comes from the previously mentioned facts, but also from the possibility of its wide application; it can be applied to achieve the following goals:

– Web site content protection (text and images displayed in a browser);
– Web site source code protection (HTML and JavaScript code);
– Prevention from Web spiders (which are obtaining e-mails found in the pages, mostly for spamming purposes);
– Web site compression, which results in faster download time, and greater Web server's overload limit.

The essence of Web Encrypt 2 is an algorithm which consists of three phases: optimization, compression and encryption. This is presented in Section 2. The process of decoding the encrypted pages in order to be displayed in a browser is described in Section 3. Additional security options are the subject of Section 4. Performances and comparison analysis are discussed is Section 5.

Let us note that official Web site for Web Encrypt is `www.design.co.yu/webencrypt/`. The latest trial version can be downloaded from `www.design.co.yu/download/v20.zip`.

2. Three-phase algorithm

The first version of Web Encrypt 2 was released a year ago, and presented in [10]. This version had one-phase algorithm (encryption). New improved version, Web Encrypt 2, is based on a three-phase algorithm – specific actions are executed in each phase. These phases, optimization, compression and encryption, are displayed in Figure 1. Boxes in this figure represent results of the former phases.

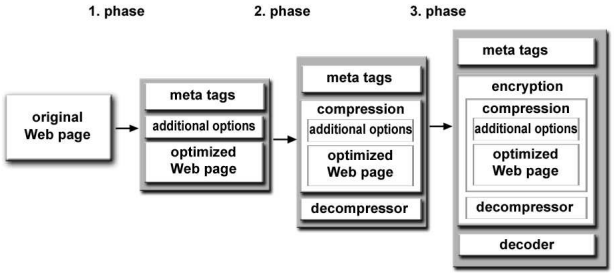

Figure 1. Three-phases algorithm

I phase - optimization

Most frequently, a Web page has a redundant code. Removing the redundant parts of the code will not effect Web page's display in browser, but will decrease its size. In this way the page is optimized. The first phase of the algorithm passes through the complete code, executing the following actions:

- META tags recognition, and its copying into buffer;
- insertion of the specified functions, e.g. disabling right mouse click function;
- recognition and elimination of the redundant parts of the code;
- recognizing the character set (code page).

META tags preserving is a very important feature of Web Encrypt 2. This action recognizes all META tags on the page and copies them to the buffer. In this way all information about the code page will be saved, making an encrypted page language independent. Information about the keywords and description will also be preserved, which is very important for search engines. While eliminating the redundant code, the algorithm can recognize the following code sequences:

- sequential space chars
- sequential new lines (\n)
- JavaScript comments (/*...*/ and / / ... \n)
- HTML comments (<- - ... - ->).

The recognition of these sequences is implemented by using the finite automates.

Web Encrypt 2 has also some additional options which increase the security level of the encrypted documents. These options are as follows:

- URL lock
- Domain lock
- disable text selection
- disable right mouse click
- disable drag & drop
- disable page printing

Each of the listed options will be discussed in detail in Section 4. All of them are implemented as JavaScript functions. The last four options are event handlers. Beside actions mentioned at the beginning of this section, code page determination is also executed in this phase. If the code page is Unicode, every character is represented by 2 bytes, opposite to ASCII characters which need 1 byte.

II phase - compression

The next phase of the three-phase algorithm is the compression. Here we will not discuss the compression in general; instead of that, for the reader's convenience we refer to the book [11].

The compression of the previously optimized page is undertaken only if the page size exceeds 5 KB. The compression of smaller pages has no effect because decompressor initially needs certain space of the page to be stored. Results achieved using the compression algorithm, named Web Pressure in this paper, show that the page size can be reduced up to 4 times, depending on the page structure.

The essence of the proposed algorithm is to find the segments of the source code which are repeated, and to write only 2 bytes (information about its position in the buffer and the length of the segment), instead of writing the whole segment.

The algorithm uses 4KB long round buffer, which means that the buffer can store up to 4096 characters. In the course of reading the page, every character is sequentially stored in the buffer (by module 4096). At the same time, the program searches for a segment in the buffer (initially 3 characters long, consisting of the $(i-2)$-nd, the $(i-1)$-st and the i-th character, where i is the position of the last read character). The length of that "window" is initially 3 because there is no saving for smaller segments. If the sought segment is found in the buffer, the window length is increased. This operation is repeated until the largest segment contained in the buffer is found. The upper limit for the window length is 19 characters. Therefore, one should take only 2 bytes (position + length, which represent segment pointer) instead of writing 19 bytes segment in the optimal case. However, the following question arises: How can the program distinguish between an actual character and segment pointer? This problem is solved by adding control signs. The control sign is an ASCII character (1 byte long) whose numerical values belong to the range (0.255). It is inserted after every eight-portioned sequence which appears as a combination of characters and segment pointers. The control signs show which members of this sequence are characters and which are segment pointers. Characters are 1 byte long, while segment pointers occupy 2 bytes. The structure of a segment pointer is shown in Figure 2.

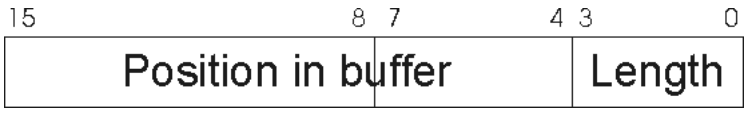

Figure 2. Segment pointer structure

The location in the buffer where the segment starts is represented by the first byte and 4 higher bits of the second byte. Since the buffer occupies 4KB, that is, 2^{12} bits, it is obvious that just 12 bits are necessary to address every

location in the buffer. Four lower bits are used for the segment length. Since the minimal length of the segment is 3 characters, it follows that the maximal length is 16+3=19 characters. For example, for the control sign 01100101, the corresponding sequence is:

0	1	1	0	0	1	0	1
char	seg	seg	char	char	seg	char	seg
1B	2B	2B	1B	1B	2B	1B	2B

The decompressor built into the page will know exactly where the segment pointers are located and how many bytes have to be read. In practice, it turns out that repeated segments in the source code appear very often, which means that the savings are reasonably large.

III phase - encryption

Encryption algorithm used in Web Encrypt 2 program is based on the substitution of characters. It resembles a trigraph algorithm which has been used for a long time in military and intelligence services for field encryption. It is worth to note that other programs of the same kind use algorithms based on the substitution of some character always by the same sign (e.g. 'a' is always replaced with 'g'), or ASCII code of a character is added with some constant. Contrary to these approaches, the proposed Web Encrypt 2 program always uses a different character in the substitution of some character. This means that the code cannot be decoded using the frequency analyzer. Furthermore, the presented program does not require that the client must have any plug-in because the decoder is built in the encrypted page; in addition, it is masked and cannot be revealed.

Our algorithm can be represented by the function f which deals with two input parameters x and y giving an output parameter z,

$$f(x, y) = z,$$

where x is a character taken from the Web page, and y is a character taken from the key – a set of characters required for the encryption. In general, the function f can be chosen arbitrary. However, the experiments have shown that a linear function gives satisfactory results. This simplification provides fast decoding, keeping at the same time the advantage of a good protection in the same time.

As mentioned above, the function f has two input parameters. The first parameter x is a character taken from the Web page that we want to encrypt. The second parameter y is taken from the specific sequence of characters (key). It is common that some part of the text or some sentence is used for the key. Characters are sequentially read from both sets. The key is always much shorter than the Web page, which means that a character from the beginning is read

again when the end is reached. Doing this, any character is substituted, but always by a different character in a new turn. Such an approach provides a uniform distribution of letters; as a consequence, there is no accumulation point, which makes the encrypted code resistant to frequency analyzers.

Let *charset* be a sequence of characters which represents all elements which are used, and let *charset.length* be the overall length of this set, often called *cardinal number*. Furthermore, let *ptr1* denote the position of a character (taken from the Web page) in the charset. Since every second parameter in the function f is sequentially taken from the key, we must determine its place in the charset; let its position be marked by *ptr2*. Then the function used in Web Encrypt 2 program can be expressed as follows:

$$f(ptr1, ptr2) = charset.length - 1 - ptr1 - ptr2.$$

As an output, the function f gives a number which represents the position of the encrypted character in the charset. The charset in Web Encrypt 2 consists of all ASCII characters (in total 256).

3. Decoding

One of the advantages of the Web Encrypt 2 program lies in the fact that a client does not need any plug-in because the decoder is built in the Web page. When the browser loads the encrypted Web page, the decoder is automatically started together with the variable that contains all encrypted data. This variable is decoded, and the result is a normal Web page displayed in the browser. But, the source code still remains encrypted.

There are two possible forms of decoder:

• Decoder is built in the Web page in the form of JavaScript code, with the size of about 200 bytes. This method is used in Web Encrypt 2. Although the decoder is also masked and there is a small possibility to be extracted from the Web page, such an approach is, to a certain extent, a weak point of this type of form. However, this shortcoming is compensated by using some additional options, especially URL or Domain lock (see §4).

• Decoder is developed in a form of Java applet. Every Web page should have only one line of code where it calls the applet, which in return performs decoding. This approach possesses two advantages – first, decoder in the applet is completely protected, and second, the presence of only one copy of the applet (decoder) on the server is quite enough; the applet will be then called by all other encrypted pages.

4. Additional options

Web Encrypt 2 contains additional options that provide further improvements concerning the Web page protection. These options are listed below:

- URL and Domain lock
- Password protection
- Disable right mouse click
- Disable text selection
- Disable drag & drop
- Disable page printing.

URL lock is an option which adds specified URL to the key during the encryption process, as shown in Figure 3.

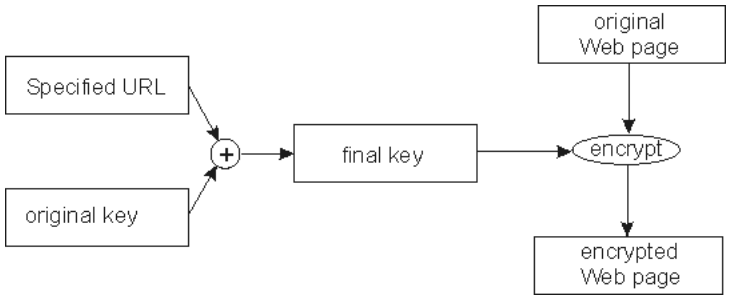

Figure 3. Encryption process with URL lock option

After decoding, the final key must be identical to the one used during the encrypting, see Figure 4. This will be the case only if the actual URL is identical to the specified URL, which means that the encrypted page must be on the specified URL. The benefit of using this option is that the encrypted page could not be saved locally, or posted on some other URL. This option compensates for the disadvantage of the first method for the first form of decoder discussed above.

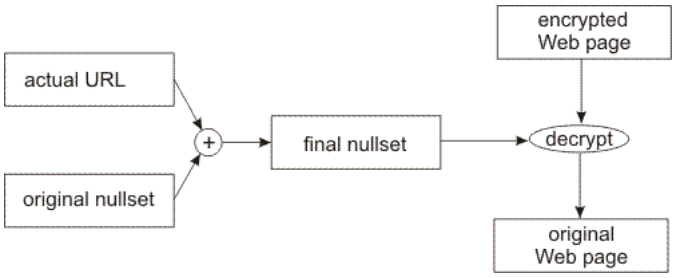

Figure 4. Decryption process with URL lock option

Domain lock is an option similar to URL lock. The basic difference is that it is not necessary to specify exact URL but only the Web domain. For example, instead of typing the following URL:

`www.design.co.yu/webencrypt/new.html`

it is enough to type

`www.design.co.yu`

In this manner, a page is protected for a specific domain, which can be useful if the exact location of the Web page is not known, or if the page location is changed within domain, or if we want to encrypt the whole web site (folder). This option works in the same manner as the previous one, except it adds the domain name to the key instead of URL.

Password protection will prompt the client who opens the page to enter a password. Only in the situation when the password is correct, the page will be decoded properly. This function is implemented in a similar way as URL/Domain lock. When someone specifies the password, it is added to the original key. The new key is then used to encrypt the page. Later, to decode the page properly, identical password must be entered because it will also be added to the original key. Obviously, the newly created key must be the same as the one used while encrypting.

Disable right mouse clicks is a very useful option, especially for the image protection. While viewing the normal (nonencrypted) page, the user can click right mouse button to open the pop-up menu with various options (Save Picture, Save Link, View Source,...). This option will cancel any effect of clicking right mouse button, disabling the user to save the image. If the user saves the encrypted page using the browser option ≫Save As − > Web Page, complete≪ (IE4+), he will save only the encrypted page without the images. This option is an event handler JavaScript function.

Disable text selection is used when someone wants to protect text content of the Web page. A user will not be able to select a text by dragging the cursor over the text. This option is an event handler JavaScript function.

Disable drag & drop, jointly with *disable right mouse clicks*, completes the image protection. It prevents a user to drag an image from the browser window into another window or desktop. This option is an event handler JavaScript function.

Disable page printing is an option intended to prevent a user to make a hard copy. Instead of the printed Web page, the user will get blank page. This option is an event handler JavaScript function.

5. Performance and comparison analysis

Web Encrypt 2 was successfully tested on the following operative systems:

- Windows 98
- Windows ME
- Windows NT
- Windows 2000
- Windows XP

Decoding time for a 206 KB long Web page was 6 sec (Athlon 650MHz, 128MB RAM, Internet Explorer 5), while decoding time for a 64 KB long Web page was 2.5 sec. Let us emphasize that our application works in all JavaScript compatible browsers (Internet Explorer, Netscape Navigator, Opera, ...).

In comparison to other programs of the same kind, Web Encrypt 2 has demonstrated better properties. For illustration, we give in this section the results of 3 tests (Examples 1, 2 and 3) performed for the comparison purpose; the output parameter is the length of the encrpyted pages for a given length of Web page. Comparison results are given in Figures 5-7.

Example 1. Web page, length: 10,000 B, Figure 5

Figure 5.

Example 2. Web page, length: 64,845 B, Figure 6

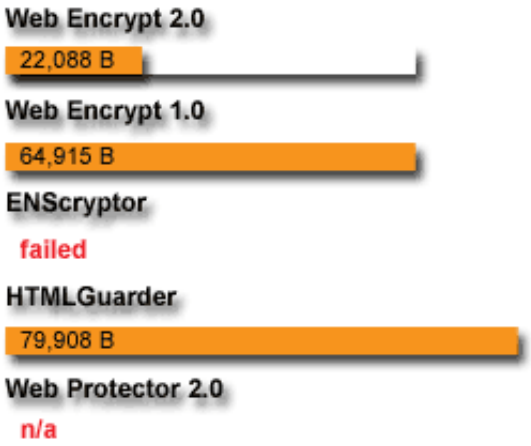

Figure 6.

Example 3. Web page, length: 211,891 B, Figure 7

Figure 7.

The results of the comparison of the presented Web Encrypt 2 with the programs of the same type, regarding some other options and features, are given in Figure 8. As usual, the marks "X", "$\surd$ " and $*$ denote the absence of validity, the presence of validity and a partial validity (respectively) of the considered feature.

main security options	disable right mouse	disable text selection	disable page printing	language independent	search engines friendly	URL lock
Web Encrypt 2.0 alpha	✓	✓	✓	✓	✓	✓
Web Encrypt 1.0	✗	✗	✗	*	✗	✓
ENScryptor	✗	✗	✗	✗	✗	✗
HTMLGuarder	✗	✗	✗	✗	✗	✗
Web Protector 2.0	✓	✓	✓	✓	✓	*
HTML Guard 2.11	✓	✓	✓	✗	✗	✗

Figure 8. Comparison analysis

6. Conclusion

Protection of the Web contents is of great importance in the field of web design. In this paper the software named Web Encrypt 2, that provides both protection and compression of web sites, is presented. The essence of the proposed program is an algorithm which consists of three phases: optimization, compression and encryption. This algorithm is a modification of a trigraph algorithm and possesses several improvements in reference to the already existing software. In comparison to other programs of the same kind, Web Encrypt 2 has demonstrated better results. Web site compression, built in Web Encrypt 2 program, provides faster download time, and greater Web server's overload limit.

Web Encrypt 2 was successfully tested on the following operative systems: Windows 98, Windows ME, Windows NT, Windows 2000, Windows XP. It works in all JavaScript compatible browsers (Internet Explorer, Netscape Navigator, Opera, ...). Web Encrypt 2 has a wide range of applications; let us mention Web site content protection (text and images displayed in a browser), Web site source code protection (HTML and JavaScript code), Web site compression, the prevention from Web spiders.

References

[1] Anderson, R., Matyas, V., Petitcolas, F., Secure books: Protecting the secure distribution of knowledge. Security Protocols Workshop, 1997.

[2] Baldin T., Bleumer, G., CryptoManager++ – an object oriented software library for cryptographic mechanisms. Proceedings of 12th IFIP International Conference

on Information Security (IFIP/Sec '96), London: Chapman & Hall, 1996, pp. 489-491.

[3] Bellare, M., Desai, A., Jokipii, E., Rogaway, R., A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 97), IEEE, 1997.

[4] Blaze, M., Schneier, B., The MacGuffin block cipher algorithm, fast software encryption. Second International Workshop Proceedings (December 1994), Springer-Verlag, 1995, pp. 97-110.

[5] Burgess, M., `http://www.iu.hio.no/ mark/lectures/security/`

[6] Deitel, H. M., Deitel, P. J., Nieto, T. R., Internet and World Wide Web: How to program. Prentice Hall, 2001.

[7] Garfinkel, S., Spafford, G., Russel, D., Web security, privacy and commerce. O'Reilly and Associates, 2002.

[8] Goldberg, I., Wagner, D., Brewer, E., Privacy-enhancing technologies for the Internet. IEEE COMPCON '97, February 1997.

[9] Landau, S., Standing the test of time: The data encryption standard. Notices of AMS, Vol. 47, No. 3 (2000), 341-349.

[10] Petković, I., Webencrypt, the program for protection of HTML, JavaScript code. Proceedings of 8th Conference on Computer Science and Information Technology YU INFO (electronic edition), Kopaonik, March 2002.

[11] Sayood, K., Introduction to data compression. Morgan Kaufman Publishers, 2000.

Received by the editors June 20, 2002.