Future Internet **2010**, *2*, 30-40; doi:10.3390/fi2010030

OPEN ACCESS

future internet

ISSN 1999-5903

www.mdpi.com/journal/futureinternet

Communication

Selective Redundancy Removal: A Framework for Data Hiding

Ugo Fiore

Università di Napoli Federico II, via Cintia 45, 80126 Napoli, Italy; E-Mail: ugo.fiore@unina.it

Received: 30 November 2009; in revised form: 4 February 2010 / Accepted: 10 February 2010 / Published: 15 February 2010

Abstract: Data hiding techniques have so far concentrated on adding or modifying irrelevant information in order to hide a message. However, files in widespread use, such as HTML documents, usually exhibit high redundancy levels, caused by code-generation programs. Such redundancy may be removed by means of optimization software. Redundancy removal, if applied selectively, enables information hiding. This work introduces Selective Redundancy Removal (SRR) as a framework for hiding data. An example application of the framework is given in terms of hiding information in HTML documents. Non-uniformity across documents may raise alarms. Nevertheless, selective application of optimization techniques might be due to the legitimate use of optimization software not supporting all the optimization methods, or configured to not use all of them.

Keywords: steganography; optimization; data hiding; covert channel

1. Introduction

The term *steganography* denotes the embedding of information (the *message*) into a digital object (the *substrate*) in such a way that the existence of the message cannot be easily discovered by anyone except the intended recipient. An additional requirement is that the semantics of the substrate is preserved and that its quality suffers (not always) as little degradation as possible [1]. Steganographic techniques can be categorized by the type of data that the substrate belongs to, such as text, images or sound [2]. A successful attack against steganography consists of simply detecting the presence of a hidden message. Awareness will suffice: decoding or removal is not necessary. Attacks against hiding techniques apply statistical analysis on the digital object or, in the case of differential analysis, they compare objects that are known to carry hidden information with objects that certainly do not.

Large files with little evident regularity, for example images and audio files, are the ideal carrier for steganography, and hiding messages in such files has indeed been thoroughly investigated. Other substrates are less appealing, but can still be used. In particular, the widespread distribution of the Web raises concerns about the usage of HTML pages as a covert communication channel, and this is even more credible in environments where low bandwidth, storage constraints, and limits in energy expenditure advise against methods that involve processing large files. As research on steganography has mainly been concentrated on images as a substrate, text is somewhat regarded as more "innocent". In fact, network administrators tend to regard HTTP traffic as legitimate, and although anti-malware systems routinely scan such traffic, these systems only look for known signatures of worms, viruses, and other members of that breed. Less attention is paid to the components of a HTML page: text, HTML tags that define page appearance, and hyperlinks towards other digital objects.

The use of natural language to conceal secret messages is known as linguistic steganography [3]. Many ideas of linguistic steganography can also be applied to computer languages, with the advantage of their formal structure. A selection of semantically neutral transformations applied to syntactic elements such as tags and identifiers can be used to conceal a message into an HTML document.

The contributions of this paper are as follows:

- a general framework, namely *Selective Redundancy Removal* (henceforth referred to as SRR), into which similar steganographic methods can be grouped and studied; instead of applying tiny additions or modifications to a substrate, SRR focuses on the elimination of existing redundancies in the substrate, striving to masquerade as an optimization technique;
- a new method, within the framework of SRR, for embedding covert messages in HTML text, based on selective optimization of HTML code and identifiers.

A proof-of-concept implementation shows the effectiveness of the proposed technique.

The rest of this paper is organized as follows. Section 2 reviews related work. The following section is dedicated to the SRR framework, a characterization of redundancies produced in HTML generation systems, and a general review of HTML optimization. In Section 4, the basic protocol for the steganographic communication is described, together with some implementation issues. Section 5 discusses various attacks and possible defenses and precedes the conclusions.

2. Related work

Encoding secret messages in web pages has been the objective of a variety of research works. Methods have included the use of invisible characters, such as blanks and tabs at the end of every line, adapting to HTML a technique used for hiding information in text files [2]. As HTML tags are case-insensitive, a combination of uppercase and lowercase can be used as a carrier for covert communication [4]. Attribute reordering has been the technique used by the author of [5], leveraging on the fact that attributes of a tag may occur in any order, and therefore a particular ordering may carry hidden information. Although redundancies play a fundamental role in steganography (see for example [6] for an interesting discussion of the usage of the metadata commonly found in proprietary file formats), to the knowledge of this author no previous work has addressed the use of selective removal of redundancies for the purpose of data hiding.

3. Selective Redundancy Removal

Owens [7] categorizes digital steganography methods into three classes, depending on the way they affect the substrate: injection, substitution, and propagation. In *injection* steganography, embedded data is inserted inside the carrier file, in places where such insertion is known to leave the semantics of the substrate unaltered. In *substitution* steganography, embedded data replaces some portions of the substrate, which are viewed as carrying insignificant information, and thus can be changed without a noticeable impact. In *propagation* steganography, a generation engine takes the payload as input and produces a phony output file masquerading as a regular content-carrying file. In all these techniques, the ability to decode the hidden message relies on some means to locate the places in the substrate where alterations are to be looked for. In the first two methods, and partly also in the third, changes are being made at specific bit locations in the substrate, identified by a key stream.

None of the three categories refers to the possible employment of transformations used in optimization: these are a class of alterations that may remove content, and nonetheless can be employed safely, because they have the property of leaving substrate semantics untouched.

3.1. Removing redundancies vs. *modifying data*

As the objective of steganography is concealing messages within apparently innocent carrier objects, efforts have focused along two main directions: increasing the capacity (the quantity of covert data that can be embedded into a cover file), and decreasing the detectability, *i.e.*, reducing the reliability of detection schemes. With the hiding techniques proposed so far, only a fraction of data can carry hidden messages without easy detection. It follows that the larger the substrate, the higher is the quantity of information which can be covertly transported. Natively large substrates such as image, audio and video files are then the ideal candidates to achieve high capacities.

On the other hand, large files need to be stored and transferred somehow, exactly as their smaller companions, requiring compression techniques to reduce data size and save on storage and bandwidth. The key principle upon which (lossless) compression lies is finding and removing statistical redundancies. Compressed files are, at least to a certain extent, redundancy-free. Having a reduced set of redundancies to manipulate, and because no value is given to the original support, which is only used as a decoy to conceal the information exchange, traditional steganographic methods have concentrated on data alteration. The most studied hiding technique, Least Significant Bits (LSB) substitution, for example, embeds secret data into an image by overwriting parts of the LSB plane. Even LSB matching, which avoids structural asymmetry [8] and is therefore not vulnerable to attacks devised for LSB substitution, essentially replaces pixel values.

If smaller substrates, especially text files, are under consideration, the requirement for compression is much less crucial, because of the diminished impact on storage and transmission resources. Consequently, information irrelevant to meaning is allowed, and frequently found, in such substrates. If not all of those redundancies are eliminated, but only a subset thereof, then a message can be embedded. This is the basic idea behind SRR. Note that even text files may become too big and involved to be easily manageable, and therefore file size or untidiness is never completely of no concern. The application of any method whatsoever that reduces file size, or simplifies processing activities such as rendering, can thus be justifiable and cannot automatically raise suspects of a concealed information transfer.

A substrate, and the process that generates it, constitute the environment defining a SRR instance. In the following, when referencing specific environments in which SRR can be applied, the process will be omitted for simplicity if it can be derived from the context.

SRR, although easily understood and applicable in the context of small files such as HTML documents, is not limited to those substrates. In fact, selective removal of redundant information can be applied to any substrate. For example, the technique can be utilized on multimedia data, by engineering modified versions of the standard compression algorithms, able to leave some amount of redundancy untouched, and masquerade that as the effect of some tradeoff between compression rate and speed. This is an open subject for future research, and this author plans to investigate the issue further.

3.2. The "reference" problem

Using SRR to hide data places its own challenges. The main reason for this becomes clear when one thinks at the decoding part of the covert transmission procedure. When the receiver detects an optimization that the sender could have made but has not, a '1' bit can safely be associated to it. On the contrary, the absence of a particular redundancy may not conclusively lead to infer a '0' bit. It may well be due to the absence of the specific redundancy in the original substrate, rather than to the omitted application of the particular optimization procedure targeted at that redundancy. It should be noted that this is not a universal problem that will arise in any environment in which SRR is used: in some environments, it is possible to unambiguously find out the subset of optimizations applied. For example, if the process is lossless compression, the substrate file, when uncompressed, will automatically yield a "reference" original that can be compressed again, analyzing the result to detect possible but unemployed optimizations. For uncompressed (and hence redundant) substrates, instead, some mechanism is called for to specify a reference. Possible workarounds to this problem are:

- the selection, or ad-hoc generation, of a substrate guaranteed to contain all of a pre-defined set of redundancies. For example, one may rely on a particular program consistently producing the same set of redundancies when being fed similar input. Such information can be obtained by prior analysis;
- transfer of the catalog of all the possible optimizations for a single substrate, thus including those that have been applied and those that have not, by other means. This "out-of-band" transmission may be done through other conventional data hiding techniques.

3.3. HTML optimization

Visual webpage editors attract both the users who are unfamiliar with the HTML language, and those who know the language but prefer the swiftness and ease of use offered by a WYSIWYG (What You See Is What You Get) editor. These programs automatically generate HTML markup that is appropriate and correct, but not necessarily tidied up and nonredundant. Common examples of this are:

- the inclusion, at the beginning of the document, of a catalog of style information, not all of which will be eventually used;

- the awkward repetition of formatting information which could have been factored up;
- the presence of formatting in places without any text at all.

All of these redundancies make a strong argument for the optimization of HTML documents, to save on storage, transfer time, and rendering time. Spiesser and Kitchen [9] describe optimization techniques that can achieve a significant reduction in the size of HTML documents: the markup in the test documents was reduced to an average of 58% of its original size.

HTML optimization consists in the elimination of redundant information, through semantically neutral transformations, *i.e.*, changes or re-arrangements that do not affect the way a document is rendered. As there are many possible such transformations that can be applied to the text, the selection of a subset of those transformations, and the extent to which such transformations are applied, can be exploited to secretly encode a message.

The use of HTML documents as a carrier for covert information exchange implies some constraints that must be met for the message not to be easily detectable. The HTML document, as displayed by a web browser, must not be affected. Every alteration is thus made on the HTML markup, and no modification is allowed on the human-readable text that the page is meant to convey, with the exception of stripping superfluous white space.

Mainly, HTML optimization focuses on HTML tags. Firstly, markup that has no effect can be safely eliminated. Re-arrangement and factoring formatting information may give very interesting savings in document size. Formatting found to be repeated over many elements could also be grouped and replaced with a style class, using Cascading Style Sheet (CSS) notation [10] that can refer to style definitions placed in the document header or in an external style sheet.

Other transformations can be done locally, *i.e.*, without taking into account the entire document structure. Some HTML tags are synonyms, in fact they have the same effect, although they not necessarily need to be of the same length. The existence of synonyms allows for multiple semantically equivalent transformations of tags. These transformations can lead to noticeable savings in size when the involved tags are particularly frequent. The resulting style mix may be the result of joining code from different sources, or a deliberate choice. Detection is thus difficult, naturally unless tag usage is inconsistent across the entire document: all the instances of the same tag in a document can be expected to uniformly adhere to a single standard.

A HTML document also contains many identifiers that label language objects such as forms fields, frames, and so on. In addition, a fair share of HTML pages include JavaScript code. In JavaScript, quite long identifiers, e.g., *current_iteration_number* may be used, although it is common programming practice to utilize very short identifiers, or even single letters such as *i*, *j*, or *k*, especially for variables used in loops. It is apparent that an identifier can be changed without impact on how the code behaves, provided that the change is applied to all the instances of the original identifier and that the new identifier is unique within its scope. If a specified list of identifiers in their long and short form is pre-shared among the parties wishing to communicate, a table-driven approach may be used to map between long and short versions. Identifiers may be also used to carry metadata information about the other global optimizations that were possible on the original substrate. The extent to which such changes are done may be used as a carrier for hidden information. Again, although inconsistencies in the use of long and short identifiers might be revealed by analysis, a mix of programming styles may also be the result of reusing and adapting someone else's code, which is a rather common practice. Thus, JavaScript identifiers are especially suited for the application of SRR techniques.

A downside in using HTML as the cover medium is that the capacity is fairly low: a single HTML file can carry a limited number of encoded bits. Therefore, if the message to be transmitted is long, several HTML documents have to be generated, although each of those documents may be small.

3.4. Scenario and possible applications

A unidirectional covert flow, resulting from a web server on the sender side and a web browser on the receiver side, can be used to transmit information between two end points, or disseminate information if the communication pattern is one-to-many. If the steganogram is split across multiple pages, the sender and the receiver(s) must agree on a prearranged sequence of web pages to reconstruct the message, although the order in which these pages are accessed is inconsequential. In another scenario, consisting in an exchange of e-mails in HTML format, the communication is bidirectional.

Possible applications include the exfiltration of small amounts of data, circumventing for example measures to limit freedom of speech and cut down on the free circulation of information. In the one-to-many scenario, an application can also be the covert coordination of a number of agents. The proposed method can be adapted to various operating environments. For example, a specialised version of the many-to-one scenario can be devised so as to reduce the resource requirements at the transmitting side, concentrating all the computation at the receiving side. This is especially attractive in resource-constrained platforms such as those typical of wireless sensor networks. Applications of the general framework should not suffer from the limitations due to the particular nature of HTML optimization and would thus cover the entire spectrum of covert communication.

4. Implementation issues

In this section, some implementation aspects are briefly discussed.

4.1. Encoding and decoding

The basic algorithms for encoding/decoding are as follows:

Figure 1. Basic algorithms for **(a)** encoding, and **(b)** decoding.

(a)	(b)
1. Generate a source HTML document known to exhibit a list of redundancies belonging to a specified set *R*. 2. Select a subset *S* among the possible optimizations *O* corresponding to *R*, according to the payload *M*. 3. Process the document with the HTML optimizer, applying only the transformations in *S*.	1. Process the document with the HTML optimizer. 2. Identify the optimization techniques that could have been applied but have not. 3. Reconstruct the message *M*.

4.2. Parsing

The first step to analyze the redundancies in a HTML document, and a pre-requisite for re-arranging its structure, factor out formatting, and in general implement all the optimization techniques outlined in the previous section, is parsing. Following [9], the idea of using the parsing engine taken from an open-source browser was abandoned, in consideration of the different requirements between a proof-of-concept implementation and a robust product used everyday by a vast community of users. Whilst the former may concentrate only of simplicity and speed, relying on the correctness of the input file, the latter must concern about graceful handling of as many exceptions and errors as possible, so a substantial portion of the code is devoted to such conditions. The presence of all this rather involved code impairs modifications.

4.3. Identifiers

A table lists some pre-shared identifier names and a list of valid abbreviations, e.g., *max_iterations* may correspond to *max_iter*, *maxIter*, or even a terse *m*, although the latter would very likely cause name clashes. The larger the number of possible synonyms, the more bits can be encoded, provided that care is taken to maintain the possibility of inferring the longest synonym that could possibly have been used for an identifier (see Section 2.1). The substrate is searched for words in the table, and the first time one such word is found, the full form or one of the abbreviated forms is chosen, according to the bits to be encoded.

4.4. HTML generation software

Here, focus will be given on methods that produce plausible sources. Different HTML editors are available, although efforts should be made towards making the particular HTML editor used not easily determinable from the output. Firstly, knowing the patterns of HTML markup generated by that system, variations in that patterns would immediately draw the attention of the adversary. Secondly, users are in general unlikely to change their HTML generation system very frequently. If the adversary were able to recognize the particular system being used, and would notice frequent changes, it would be a strong warning of the possible presence of a hidden message. Therefore, some pre-processing is needed, in order to eliminate signatures and the characteristics of the particular HTML editor that may lead to its individuation. A less valid alternative could be, if the generation program provides a rich set of user-definable options affecting its behavior, to frequently change these options.

4.5. Test data

To verify the method effectiveness against real web pages, a sample of pages actually visited on a live network is needed. For this purpose, the cached files of the campus web proxy server running *squid* [13] have been analyzed and extracted with the help of the *squidpurge* tool [14]. Of a total of 96134 unique URLs, 10118 were found to refer to HTML documents (*i.e.*, excluding other types of data, e.g., images or Javascript include files). File sizes ranged from 220 bytes to 3.3 Mbytes, with an average of 22478 bytes. Exploitable redundancy could be found in 38.7% of the files. The embedding capacity was in the range of 0.07-0.1%, depending on the selection of parameters, with more conservative choices producing lower values.

5. Discussion

Various attacks can be brought against steganographic encoding methods. Each data hiding technique that is published gives rise to a series of papers describing attacks that are successful against it, and each of those attacks, in turn, stimulates the development of appropriate countermeasures [11,12].

The adversary is assumed to know about the existence of the basic protocol for information embedding, but to have no access to the configuration of the SRR system, *i.e.*, all the parameters that influence the encoding/decoding, including the choice of a catalog of optimization with their associated thresholds, pre-processing and post-processing operations, and dictionaries. All this information is assumed to have already been shared (by means of some unspecified protocol) between the parties wishing to communicate covertly. As with most steganographic systems, the hidden message itself can be encrypted with a secret key, making it harder for the adversary to perform guessing attacks on the secret message (as this message will result in a pseudorandom bitstream).

In order to show that a message is hidden by SRR in a HTML document, the adversary would have to prove that the partially optimized page could not be plausibly generated by ordinary optimization, by an HTML generation program, or even by hand. Even if the adversary had reasonable grounds to suspect that selective optimizations had been done, the adversary would still have to prove that this behavior is not simply the result of a legitimate choice of user-definable options in the HTML generation program, in some optimization program, or both.

The adversary might look for internal inconsistencies in the document. Simultaneous use of long and short tags, for example, can be interpreted as a signal that selective optimization was applied. However, the base of HTML documents shows that such internal inconsistencies are frequent and depend on the configuration of the generating program, on user preferences, or on programming style. Thus, it would not be enough for the adversary to mark as suspicious all the pages that do not correspond exactly to a few patterns of tag use. The rate of false positives, thus, can be quite high. Analogously, the adversary cannot flag as suspicious all HTML documents that do not exactly match the redundancy pattern of known HTML editors. There are three reasons for this. The first is that there are many tools that generate HTML code, and each of them must be taken into account. The second is that each of these systems may offer its users many options that affect the redundancy patterns in the output. The third is that some of these tools are open-source and are thus amenable to minor or substantial modifications in their behavior, as the result of amendments to the source code done by someone wishing to adapt the program to their particular requirements. Needless to say, there is no reason why such customized systems must be made public.

5.1. Substrate secrecy

Disclosing the original, clean, carrier files or using them more than once can be dangerous, as a differential analysis attack could reveal the presence of a hidden message. A fresh, secret substrate must be chosen or preferably created, possibly collating text from pages that are continuously updated such as news. In order to take advantage of a combination of different styles, pieces of text originated from multiple sources can be combined. For instance, some different articles about the same theme could be concatenated. Keeping the original formatting could then be justified with willingness to get the original components clearly separated.

5.2. Evolution of HTML generation systems

The steganographic encoding presented here, being based on selective elimination of existing redundancies, is based on the imperfection of HTML generation systems, in the sense that it relies on the presence of bulky and redundant HTML markup in the documents generated by those systems. It is therefore vulnerable to the progress of HTML generation technology: if generators will increase their ability to produce code that is tidier and more slender, the amount of redundancy in the resulting documents might vanish or become too small to be amenable data hiding by SRR.

5.3. Active wardens and watermarking

SRR of HTML documents is vulnerable to the attacks of an active warden, *i.e.*, a warden who is able to alter the content of the exchanged messages, leaving their semantics unchanged (a warden who is only able to inspect the messages in transit is said to be passive). By randomly adding useless information, an active warden may disrupt the hidden communication. However, it should be kept in mind that sensitivity to alterations performed by active warden is quite common amongst text-based embedding methods. In such variants of steganography, a low detectability of the embedding is more of a concern than the robustness against modifications by an adversary. The adversary is supposed to be passive and only interested in spotting the covert communication.

5.4. Use of HTML optimization software

The same argument of the previous sections applies to the diffusion of HTML optimization programs. If the use of such programs becomes common practice, HTML pages would be natively free of the redundancies which are a prerequisite for SRR, making the technique not viable.

5.5. Identical portions problem

The necessity of a uniform treatment of identical portions of HTML markup has already been brought up elsewhere in this paper. Here, it is important to underline that this requirement for uniformity is much stronger for “local” substitutions (e.g., short tags for longer ones) than it is for “global” rearrangement and factoring of formatting information. The simultaneous use of different tags with the same meaning would be a strong warning, as the only plausible reason for this would be that the document was the result of a merge of different parts generated by different programs. The different level of factoring, instead, might be the result of the sequence of operations (additions, deletions, copy-and-paste) that led to the HTML page in its current form. This weakness is further alleviated by the design of pages containing separate portions, each with a specific formatting.

Differential analysis applied to similar portions of HTML documents generated at subsequent moments in time is also a powerful tool in the hands of an adversary that would be able to show how similar portions of formatting are handled in a different fashion. Defense is based on the selection of the original substrates, paying as much attention as possible to the avoidance of very close repetitions. The adversary may also monitor web activity strictly, eventually building up a good knowledge base about the characteristics of HTML documents that were previously generated by each source. The sources should therefore be often changed.

6. Conclusions

This paper introduces a novel data hiding scheme, Selective Redundancy Removal (SRR), based on the selective application of a subset of semantically neutral transformations, chosen among those commonly used when optimizing data for reducing size and increasing processing speed. The receiver may decode the message by looking at the optimizations that the sender could have carried out but had not. An example of the application of the SRR framework is hiding of information in a HTML document. Visual web page editors are known to produce, for the sake of their own efficiency and speed when being run, bulky and inefficient HTML markup, with repetitions and code having no effect. Removal of this unneeded markup, if performed selectively, can be used to hide a message. An adversary would have to prove that the selectively optimized code would unlikely have been produced by a HTML generation system, by an optimization system, by hand, or a combination thereof. The proposed technique has an admittedly low relative capacity as compared to LSB-like approaches. Differently from those encodings, SRR does not require an encoding sequence that determines the locations in the carrier where encoded bits must be found. This would be especially noticeable with non-textual supports. Finally, LSB (Least Significant Bit) steganographic techniques traditionally employed on images and audio affect the source with tiny alterations, relying on the perceptual error margin. In contrast, like lossless data embedding methods (see for example [15]) SRR does not destroy any semantically-influent information in the source, as only optimizing transformations are used.

References and Notes

1. Katzenbeisser, S.; Petitcolas, F., Eds. *Information Hiding – techniques for steganographic and digital watermarking*; Artech House: Norwood, MA, USA; 2000.
2. Bender, W.; Gruhl, D.; Morimoto, N.; Lu, A. Techniques for data hiding, *IBM Syst. J.* **1996**, *35*, 313-336.
3. Taskiran, C.M.; Topkarab, U.; Topkarab, M.; Delp, E.J. Attacks on Lexical Natural Language Steganography Systems. In *Proceedings of the SPIE international conference on Security, Steganography, and Watermarking of Multimedia Contents (SPIE 6072)*, San Jose, CA, USA, January 2006; Delp, E.J., III, Wong, P.W., Eds.; pp. 97-105.
4. Sui, X.G.; Luo, H. A New Steganography Method Based on Hypertext. In *Proc. of Asia-Pacific Radio Science Conference,* Qingdao, China, August 2004; Keyun, T., Dayong, L., Eds.; IEEE Computer Society Press: Los Alamitos, CA, USA; pp. 181–184.
5. Ghosh, S. *StegHTML: A message hiding mechanism in HTML tags*. Technical report, Univ. of Virginia, 2007; http://www.cs.virginia.edu/~skg5n/main.pdf. Accessed on 21 January 2010.
6. Castiglione, A.; De Santis, A.; Soriente, C.: Taking advantages of a disadvantage: Digital forensics and steganography using document metadata. *J. Syst. Software* **2007**, *80*, 750-764.

7. Owens, M. *A discussion of covert channels and steganography*, SANS Report, 2002.
8. Sur, A.; Goel, P.; Mukhopadhyay, J. A Novel Steganographic Algorithm Resisting Targeted Steganalytic Attacks on LSB Matching, in *Digital Watermarking: 7th International Workshop (IWDW 2008), Busan, Korea, November 10-12, 2008, Selected Papers*, Springer-Verlag: New York, NY, USA, 2009; pp. 199-208.
9. Spiesser, J.; Kitchen, L.; Optimization of HTML automatically generated by WYSIWYG programs. In *Proceedings of the 13th international conference on World Wide Web*, New York, NY, USA, May 2004; Feldman, S.J., Uretsky, M., Najork, M., Wills, C.E., Eds.; ACM Press, New York, NY, USA; pp. 355-364.
10. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, http://www.w3.org/TR/CSS2 Accessed on 21 January 2010.
11. Chen, Z.; Huang, L.; Yu, Z.; Li, L.; Yang, W. A Statistical Algorithm for Linguistic Steganography Detection Based on Distribution of Words. In *Proceedings of the Third International Conference on Availability, Reliability and Security (ARES 2008)*, Barcelona, Spain, March 2008; Jakoubi, S., Tjoa, S., Weippl, E.R., Eds.; IEEE Computer Society Press: Los Alamitos, CA, USA; pp. 558-563.
12. Huajun, H.; Xingming, S.; Zishuai, L.; Guang, S. Detection of Hidden Information in Webpage, *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, 2007; Volume 4, pp. 317-321.
13. Squid, http://www.squid-cache.org/. Accessed on 21 January 2010.
14. squidpurge, http://www.wa.apana.org.au/~dean/squidpurge/. Accessed on 21 January 2010.
15. Fridrich, J.; Goljan, M.; Du, R. Lossless data embedding-new paradigm in digital watermarking. *EURASIP J. Appl. Signal Process.* **2002**, *2*, 185-196.