The Journal of Systems and Software 85 (2012) 1075–1082

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

UniSpaCh: A text-based data hiding method using Unicode space characters

Lip Yee Por*, KokSheik Wong, Kok Onn Chee

Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia

ARTICLE INFO

Article history:
Received 6 March 2011
Received in revised form
14 September 2011
Accepted 9 December 2011
Available online 17 December 2011

Keywords:
UniSpaCh
DASH
Data hiding
Unicode character
Space manipulation

ABSTRACT

This paper proposes a text-based data hiding method to insert external information into Microsoft Word document. First, the drawback of low embedding efficiency in the existing text-based data hiding methods is addressed, and a simple attack, DASH, is proposed to reveal the information inserted by the existing text-based data hiding methods. Then, a new data hiding method, UniSpaCh, is proposed to counter DASH. The characteristics of Unicode space characters with respect to embedding efficiency and DASH are analyzed, and the selected Unicode space characters are inserted into inter-sentence, inter-word, end-of-line and inter-paragraph spacings to encode external information while improving embedding efficiency and imperceptivity of the embedded information. UniSpaCh is also reversible where the embedded information can be removed to completely reconstruct the original Microsoft Word document. Experiments were carried out to verify the performance of UniSpaCh as well as comparing it to the existing space-manipulating data hiding methods. Results suggest that UniSpaCh offers higher embedding efficiency while exhibiting higher imperceptivity of white space manipulation when compared to the existing methods considered. In the best case scenario, UniSpaCh produces output document of size almost 9 times smaller than that of the existing method.

1. Introduction

Data hiding is the art and science of inserting payload (external information) into a host content (Wu and Liu, 2002). Earlier information hiding methods merely embed payload into a cover (e.g., text document, image and audio), and in recent years, specialized data hiding methods are proposed to serve specific purposes. For instance, in steganography, the cover content is carefully manipulated to encode payload while aiming to conceal the very existence of the encoded information. In application of watermarking, copyright information of a content is inserted into the content itself to claim ownership in case of a dispute. Other applications include annotation, indexing, error correction, etc. where more information could be found in Katzenbeisser and Petitcolas (2000).

Although various multimedia contents are utilized as the host to embed payload in recent years (Wu and Liu, 2002; Xuan et al., 2007), text documents remain to be an important choice of host due to its ubiquitous existence in the digital domain. Rightly or wrongly, many data hiding methods (Atallah et al., 2001; Topkara et al., 2005) are used for document tracking or copyright protection. Others are used for authentication (Atallah et al., 2003), and recently, for steganography (Gutub and Fattani, 2007; Liu and Tsai, 2007; Por et al., 2008). Loosely speaking, text-based data hiding method in the digital domain can be divided into semagrams and open codes (Kessler, 2004). Semagram embeds payload by means of changing the appearance of the host-text such as adding extra spaces or tabs while using some pre-defined data representation (e.g., one space character →'0' and two space characters → '1'). The open space method proposed by Bender et al. (1996) is one of the earliest instance of text semagrams. Here, the secret message is encoded by manipulating white spaces in a document in which case inter-sentence, inter-word and end-of-line spacings are considered. Other semagram approaches could be found at Por et al. (2008), Khairullah (2009), Kwan (2006), wbStego (2004) and Chotikakamthorn (1998).

Obviously, one can also treat text document as image where the existing image processing based data hiding methods can be readily applied (Yu et al., 2005). Nevertheless, there are methods that manipulate space in a document image, at pixel level, to embed information and they are robust against the print and scan processes. For example, He et al. (2009) divide texts into $N \times N$ blocks where N is an integer and for each block, they consider the ratio of *black pixel* to *total number of pixels in a block* as the characteristic value. These characteristic values ($N \times N$ altogether) are transformed into the frequency domain using two dimensional discrete cosine transformation. The authors found that the processes of printing then scanning will only increase the DC value while the AC values remain mostly unchanged and hence high frequency components are modified to embed payload. On the other hand, Zou

* Corresponding author.
E-mail addresses: porlip@um.edu.my (L.Y. Por), koksheik@um.edu.my (K. Wong), wingz_04@yahoo.com (K.O. Chee).

0164-1212/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2011.12.023

and Shi (2005) divide a row of texts in two sets of word (say ϕ_A and ϕ_B) so that the sum of inter-word spaces (i.e., the number of pixels between two consecutive words) in both sets is approximately the same. Inter-word spaces in each set are either contracted or expanded such that certain conditions are satisfied to encode information. That is, $|\phi_A| > |\phi_B| + \epsilon$ is imposed to embed '0', and vice versa, where $|\phi_B|$ is the total number of inter-word spaces in set ϕ_B and $\epsilon > 0$. Culnane et al. (2006) further improve this work by using multi-set modulation technique to achieve higher carrier capacity.

On the other hand, open codes rely heavily on the pre-defined extraction/decoding method. Basically, a paragraph is prepared in a way so that it appears to be innocuous to visual attack. The embedded secret message is only revealed when the grille/template is superimposed on the paragraph to mask out texts used merely for camouflaging purposes. A classical example of open codes (i.e., null cipher) is as follows:

> Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils (Johnson and Jajodia, 1998).

When the second letter is extracted from each word, it reveals the message of *Pershing sails from NY June 1*.

In addition to semagram and open codes, NLP (natural language processing) is also an important branch of data hiding in the text domain. Probabilistic context-free grammar, synonym substitution, natural language generation, text paraphrasing, and sentence tree structure manipulation are some of the common techniques in NLP-based data embedding (Topkara et al., 2005). For example, synonym substitution methods are proposed to embed payload by replacing words which have similar meaning (Calvo and Bolshakov, 2004; Topkara et al., 2006). Abbreviation of word (e.g., 'i.e.') or phrase and its corresponding unabridged version (e.g., 'that is') are considered in pair to embed information (Shirali-Shahreza and Shirali-Shahreza, 2007). Sentence structure manipulation method such as shifting the location of the noun and verb was proposed by Murphy and Vogel (2007) to embed payload. Later, Atallah et al. (2003) proposed semantic transformation method to embed data by adding or removing repeated information of a particular content. Hybrids of the aforementioned methods are also proposed and more information could be found at Nakagawa et al. (2001), Niimi et al. (2003), Bergmair (2004), Bergmair and Katzenbeisser (2004, 2007) and Chand and Orgun (2006). Interesting ways of relating image watermarking to NLP watermarking are presented by Topkara et al. (2005). Recently, the 'change tracking' feature in word processor is exploited to encode payload. In particular, Liu and Tsai (2007) mimic the product of a collaborative writing effort. In the mimicking process, a Microsoft Word document is degenerated to form the draft of itself, and the degenerated document is manipulated to embed information, faking that another author is editing the document.

In this paper, we focus on data hiding method based on space character manipulation (i.e., open space method). A novel text-based data hiding method UniSpaCh is proposed to embed information in Microsoft Word document using Unicode space characters. Microsoft Word document is considered as the host because 80% of the enterprise customers are using Microsoft Office for worker productivity and collaboration (Montalbano, 2009). In addition, white spaces are considered to encode payload because they appear throughout the document (i.e., available in large number), and the manipulation of white spaces has insignificant effect to the visual appearance of document. UniSpaCh embeds payload into inter-sentence, inter-word, end-of-line and inter-paragraph spacings by introducing Unicode space characters. Experiments are carried out to verify the performance of UniSpaCh as well as comparing it to the existing text-based data hiding methods.

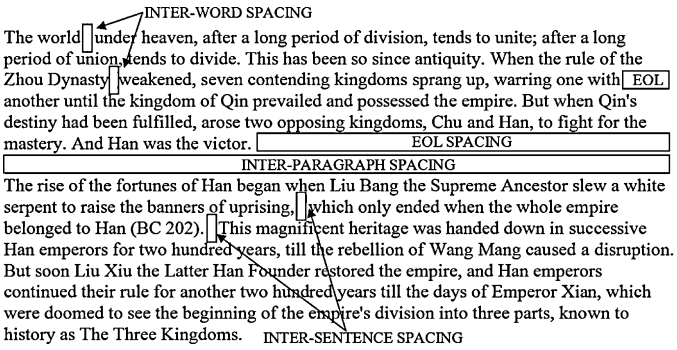

Fig. 1. Illustration of inter-word spacing, inter-sentence spacing, EOL (end-of-line) spacing and inter-paragraph spacing.

2. Related tools

For the rest of the discussion, let *inter-sentence spacing* denotes the spacing between two consecutive sentences (including those separated by comma), and let *inter-word spacing* denotes the spacing between two consecutive words. Also, let *end-of-line spacing* refers to the remaining spaces of a line after the terminating character (e.g., period, semicolon) that follows the last word in a paragraph. The empty line between two consecutive paragraphs is referred to as *inter-paragraph spacing*. The aforementioned spacings are illustrated in Fig. 1. Define embedding efficiency as the ratio of *number of embedded bits* to *output filesize*. The following subsections briefly describe the existing open space data hiding methods that exploit the aforementioned spacings to encode payload. Note that there are other existing methods that hide payload directly into the header of a document/text file, using different feature encodings (e.g., color, shape, size), etc. but they are not categorized as open space data hiding method where white space is the only entity considered and manipulated to encode the payload. Hence, we only consider methods where encoding and decoding of the payload take place directly in the text document without pre-processing such as document-to-image conversion.

2.1. SNOW: encode as space

SNOW is a steganographic tool developed by Kwan (2006) where the payload is concealed by appending white space characters at end-of-line spacing. A character from the message is encoded as a triplet of bits and embedded by introducing 0–7 spaces (Por et al., 2008; Kwan, 2006). Here, the tab space character is introduced to delimit the triplet-bits. For the same output document file size, SNOW offers higher carrier capacity than Bender's methods (Bender et al., 1996). When the cover text is too short to host the entire payload, SNOW embeds the remaining payload at the end of the cover text by using end-of-line spacing.

2.2. Spacemimic

Spacemimic considers the end-of-line spacing and inter-paragraph spacing to embed payload (McKellar, 2000). Spacemimic utilizes a single white space character to represent '0' and a tab space character to represent '1'. Similar to SNOW, end-of-line spacing/inter-paragraph spacing is utilized to host the remaining payload.

2.3. wbStego4open

wbStego4open utilizes the mixture of inter-sentence spacing and inter-word spacing to embed payload. White space character is substituted by an encoding value of 0x00 to embed '1', or an

encoding value of 0x20 to embed '0' (Murphy, 2001). In terms of capacity, wbStego4open is able to embed more information when compared to SNOW (with an average of 1 bit per eight bytes of cover-text) if size of the output document is limited to some constant. Unlike SNOW, wbStego4open ensures that the cover-text is long enough to host the entire payload before the embedding process takes place.

2.4. WhiteSteg

Por et al. (2008) invented WhiteSteg that embeds secret message by using the mixture of both inter-word spacing and inter-paragraph spacing. A single white space character denotes '0' and two white space characters denote '1' of the message. When a cover text is not long enough to host the payload, texts from the original cover document will be copied and appended to the output document, word by word, until there are just enough texts to host the payload.

To fully utilize the white spaces in a document, tab characters are appended to the end of a sentence or between paragraphs as delimiters to ensure that the number of spaces introduced can be counted correctly, which leads to successful decoding of the embedded payload. When limiting the output document size to a constant, WhiteSteg offers higher embedding capacity when compared to SNOW, wbStego4Open, as well as Spacemimic.

3. Drawbacks of existing methods and dot and arrow attack (DASH)

Although payload can be embedded using white spaces, the aforementioned tools/approaches require sufficiently large cover-document to hide merely a few bits of the payload. This is attributed to the dependency of carrier capacity on the number of available white spaces in the document. For instance, WhiteSteg (Por et al., 2008) can merely host approximately ~13 bits of the payload in an empty line (i.e., inter-paragraph spacing) using 12pt font size of Courier New, which is not viable to conceal a relatively large amount of payload.

In addition, since all space manipulating data hiding methods introduce new spaces (i.e., space character and/or tab space character) into the cover document, irregular space count between words/sentences or awkward appearance of tab raises suspicion. If one marks the space character by a rarely utilized symbol and the tab character by another symbol, the existence of embedded payload in the document is instantly revealed. We name this simple visual attack as DASH (dot and arrow show/hide). In particular, since most of the word processors are equipped with the "show or hide formatting marks" (Microsoft Words© 2007) or "formatting aids page" (OpenOffice Writer) feature, we can use this feature to screen for manipulated documents and attempt to extract the embedded information in case a document is deemed suspicious. In the case of Microsoft Word and Writer, a space is denoted by '·' and a tab is represented by '→'. Therefore, one just needs to observe the frequency of '·' and '→' as well as the location at which they appear. Moreover, if a document is deemed suspicious, the information embedded by the existing methods considered can be readily decoded because simple data representation scheme (i.e., one space → '0' and two spaces → '1' or the opposite) is utilized. Fig. 2 shows the formatting marks in Microsoft Word 2007 for document manipulated by SNOW, Spacemimic, wbStego4Open and WhiteSteg. In all four cases, the spaces or tabs introduced are clearly visible, suggesting that the documents are manipulated. Fig. 2(b) shows that wbStego4Open also introduces other character (i.e., Unicode character of ÿ) in addition to the ordinary space, which motivated our data hiding method using Unicode space characters.

Table 1
Unicode space characters.

Code	Name
U+0020	Space
U+00A0	No-Break Space
U+1680	Ogham Space Mark
U+180E	Mongolian Vowel Separator
U+2000	En Quad
U+2001	Em Quad
U+2002	En Space
U+2003	Em Space
U+2004	Three-Per-Em Space
U+2005	Four-Per-Em Space
U+2006	Six-Per-Em Space
U+2007	Figure Space
U+2008	Punctuation Space
U+2009	Thin Space
U+200A	Hair Space
U+202F	Narrow No-Break Space
U+205F	Medium Mathematical Space
U+3000	Ideographic Space

4. Unicode space character-based data hiding (UniSpaCh)

Similar to the existing data hiding methods (Por et al., 2008; Kwan, 2006; wbStego, 2004; McKellar, 2000), our method UniSpaCh considers the mixture of inter-sentence, inter-word, end-of-line and inter-paragraph spacings to embed information into Microsoft Word document. However, to withstand DASH attack while aiming to increase embedding efficiency, different set of Unicode space characters is utilized, depending on the type of spacing. In general, there are 18 space characters in Unicode Standard Version 5.2 (Allen et al., 2009) and they are tabulated in Table 1. After a simple verification with the show/hide formatting marks feature in Microsoft Word 2007 running on several versions of Microsoft Windows, we conclude that only eight Unicode space characters, namely, En Quad, Em Quad, Three-Per-Em, Six-Per-Em, Figure, Punctuation, Thin and Hair, are suitable for data hiding. In this context, a space is said to be suitable for data hiding if it appears as an imperceptible space (i.e., without any mark/sign indicating its existence) with respect to DASH. The rest of the ten Unicode space characters are inevitably revealed as square or degree symbol under DASH attack.

The relative spacing/width occupied by the eight selected Unicode space characters are shown in Fig. 3. The results suggest that there are space characters (e.g., Em Quad) that occupy larger width than that of the ordinary space character, and vice versa (e.g., Hair). For that, these characters will be utilized differently, depending on the type of spacing under consideration. In particular, the spacings in a Microsoft Word document are classified into two groups, namely (A) inter-words spacing and inter-sentence spacing, and (B) end-of-line and inter-paragraph spacing. For Group A, if the ordinary space characters are replaced by any of the selected eight Unicode space characters (to encode 3 bits of information since there are 8 of them), the distance between two words or two sentences is either too wide or too narrow, which raises suspicion. In addition, since the Unicode space characters will not appear as '·' nor any other symbols (i.e., invisible) under DASH, it will also raises suspicion. Therefore, instead of the simple replacement approach, three Unicode space characters that occupy the smallest width are selected and combined with the ordinary space character to encode payload in segments of 2 bits. An example of the data representation scheme is shown in Fig. 4 in which case their order could be permuted using secret key to further complicates unauthorized decoding of the embedded information. The corresponding combination of ordinary space and Unicode space character will be chosen depending on the information to be embedded.

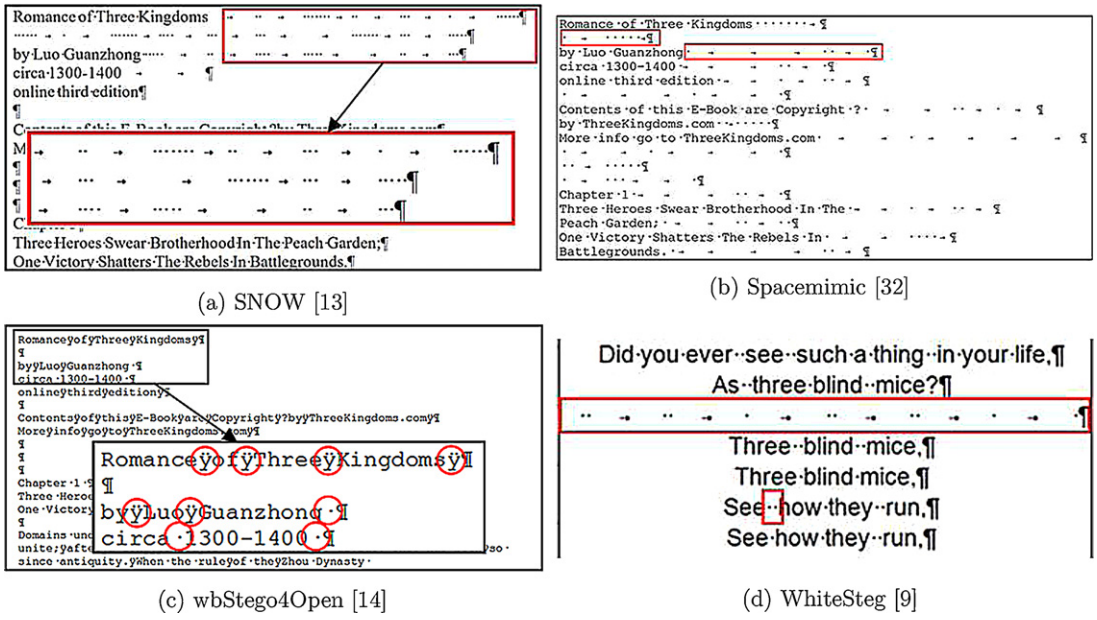

Fig. 2. Space character '·' and tab character '→' are revealed by using "show/hide formatting marks" feature in Microsoft Word 2007.

Space Character	Windows XP Hide	Windows XP Show	Windows Vista Hide	Windows Vista Show	Windows 7 Hide	Windows 7 Show
Space	abc def	abc·def	abc def	abc·def	abc def	abc·def
En Quad	abc def	abc def	abc def	abc def	abc def	abc def
Em Quad	abc def	abc def	abc def	abc def	abc def	abc def
Three-Per-Em	abc def	abc def	abc def	abc def	abc def	abc def
Six-Per-Em	abc def	abc def	abc def	abc def	abc def	abc def
Figure	abc def	abc def	abc def	abc def	abc def	abc def
Punctuation	abc def	abc def	abc def	abc def	abc def	abc def
Thin	abc def	abc def	abc def	abc def	abc def	abc def
Hair	abc def	abc def	abc def	abc def	abc def	abc def

Fig. 3. Ordinary space character and Unicode space characters that are invisible with respect to the show/hide formatting marks feature in Microsoft Word 2007.

For Group B, if the ordinary space characters and tabs are utilized in combination to encode payload (as in WhiteSteg, Por et al., 2008; SNOW, Kwan, 2006; Spacemimic, McKellar, 2000), existence of the embedded information is immediately revealed using DASH. To avoid the appearance of '·' and '→', we may utilize all eight selected Unicode space characters to embed 3 bits for every space considered. However, some Unicode space characters occupy larger width than others, the choice of Unicode space characters will affect the actual carrier capacity. For that, a simple analysis on two cases is performed. In case (i), all eight Unicode space characters are considered where each character encodes a three-bit sequence. In case (ii), the four Unicode space characters that occupy the smallest width, namely Hair Space, Six-Per-Em Space, Punctuation Space and Thin Space, are considered where each space character encodes a two-bit sequence. A random sequence of zeros and ones are embedded using case (i) into an inter-paragraph line until the entire line is occupied, and the number of Unicode space characters introduced is counted. These procedures are repeated using case (ii). The results reveal that ~105 and ~287 Unicode space characters can be fitted into an inter-paragraph line in case (i) and case (ii), respectively. In other words, when using case (i), ~315 bits are embeddable into an inter-paragraph line and using case (ii), ~574 bits can be embedded into the same line. Therefore, case (ii), i.e., Hair Space, Six-Per-Em Space, Punctuation Space and Thin Space, are utilized for data embedding in UniSpaCh for handling Group B. An example of the external data representation scheme is shown in Fig. 5.

Symbol	Combination	Sequence
	Normal	00
	Thin + Normal	01
	Six-Per-Em + Normal	10
	Hair + Normal	11

Fig. 4. Representation scheme for inter-word spacing and inter-sentence spacing (Group A).

Symbol	Characters	Sequence
	Hair	00
	Six-Per-Em	01
	Punctuation	10
	Thin	11

Fig. 5. Representation scheme for end-of-line and inter-paragraph spacings (Group B).

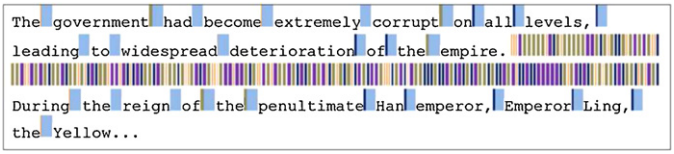

Fig. 6. Output example of UniSpaCh with the Unicode space characters color-coded.

5. Discussions

Fig. 6 shows an output example of Microsoft Word document manipulated by UniSpaCh, with the Unicode space characters color-coded. Unlike the existing text-based data embedding methods that reveal the existence of the inserted information when analyzed by DASH, output product of UniSpaCh does not show traces of manipulation such as awkward '·' or '→' throughout the document. Furthermore, in the case of wbStego4Open, ÿ appears at awkward locations when analyzed by DASH because the encoding value of 0x00 is utilized to store '1'. Even if the existence of embedded information is known to an attacker, UniSpaCh is more robust against unauthorized decoding of the embedded information. This is because UniSpaCh encodes two bits per combination as shown in Figs. 4 and 5.

Theoretically, UniSpaCh is able to achieve higher embedding efficiency when compared to the existing methods. Here, we consider the unit of bps (bits embedded per space introduced). For spacings in Group A, the existing methods encode, on average, 2/1 = 2 bps since $P(w = \text{'0'}) = P(w = \text{'1'}) = 0.5$ and space is only introduced when $w = 1$. In UniSpaCh, since 3 combinations out of 4 (refer to Fig. 4) introduce a Unicode space character, it encodes, on average, $2/0.75 \sim 2.67$ bps, which is higher than that of the existing methods. For spacing in Group B, the existing methods use tab as the delimiting character to separate sequence of space characters, which significantly reduces the number of space characters that can be put into a line. On the other hand, instead of using tab which occupies large space (i.e., width) in a line, UniSpaCh uses Hair Space, Six-Per-Em Space, Punctuation Space and Thin Space in which case each of them occupies a smaller width than that of an ordinary space character. In addition, each Unicode space character in UniSpaCh encodes 2 bits of the payload. Therefore, for a given text document (without repeatedly copying and appending), we expect UniSpaCh to offer higher embedding capacity when compared to the existing methods considered.

Next, we do not deny that there is a possibility that the existence of the payload (hidden by using UniSpaCh) can be detected if someone launches statistical analysis aiming only at Unicode characters. However, there are many technical challenges to automate an attack because such an attack must be able to discriminate open space data hiding method from other approaches such as NLP, feature manipulation and header modification. Also, the automated attack must be able to parse the suspicious document (e.g., Microsoft Word, OpenOffice Write, iWork Pages) correctly for further analysis, which is even more difficult especially when the embedding rate is low. In case the algorithm overcomes the aforementioned obstacles and attempts to extract the embedded information, it is not straight forward either because the payload can embedded by using encoding format such as binary, hexadecimal and ASCII. Nevertheless, to prevent the hidden information from being extracted by unauthorized viewer or attacker, there are at least two approaches, namely:

1. *mapping function* – the association of message bits shown in Figs. 4 and 5 can be changed periodically or non-periodically using a secret key (i.e., seed to the pseudo-random number generator) to further complicate unauthorized extraction of the embedded information. In particular, one of the 4 ! = 24 mappings can be utilized for encoding/decoding the payload for some fixed or variable length of the message, and then a different mapping is applied.
2. *extra security layer* – the payload is encrypted (say using AES or DES) prior to data embedding. For such implementation, even if the encrypted bits are revealed, the attacker still needs to crack the encryption layer.

Last but not least, since this is a preliminary study of UniSpaCh and its performance, we consider visual inspection as the representative attack. The visual attack DASH is proposed because it can be carried out using the built-in function of any recent word processing software. The human visual system is trained to recognize known entities and hence this ability is exploited for visual attacks (Westfeld and Pfitzmann, 1999). In open space information hiding method, robustness against visual attack is the first issue that needs to be addressed, analog to visual inspection in image based data hiding method. An open space data hiding method is considered vulnerable to visual attack if one can, by naked eyes, determine the existence of payload embedded within the document in question. Since "show or hide formatting marks" (MicrosoftWords© 2007) or "formatting aids page" (OpenOffice Writer) are readily available at a few mouse clicks, even a layman can carry out the proposed DASH attack. Thus, an attacker can screen for manipulated documents and attempt to extract the embedded information. For that, we focus on countering the proposed visual attack DASH. The investigation on securing the payload from other attacks such as automated statistical analysis will be considered as part of our future work.

6. Experiments

First, it is verified that the payload inserted by the proposed method UniSpaCh can be entirely decoded. The inserted Unicode space characters could be removed to completely reconstruct the original document, and hence the reversible functionality is verified. To the best of our knowledge, all text-based data hiding methods using open space are limited to SNOW, Spacemimic, wbStego4Open, and WhiteSteg. Since the proposed method UniSpaCh can achieve both encoding and decoding of payload directly with the text document, to have a fair comparison, we only considered existing methods that offer the same features. Other methods such as Chen et al. (2006) that require text-to-image conversion for encoding/decoding of payload (but not in the text document itself) is not considered in this context.

Secondly, the robustness of UniSpaCh with respect to DASH is verified. Fig. 7 shows an output example of Microsoft Word document manipulated by UniSpaCh. Here, the show/hide formatting marks are turned on to mark the ordinary space and tab as '·' and '→', respectively. It is observed that there is no awkward

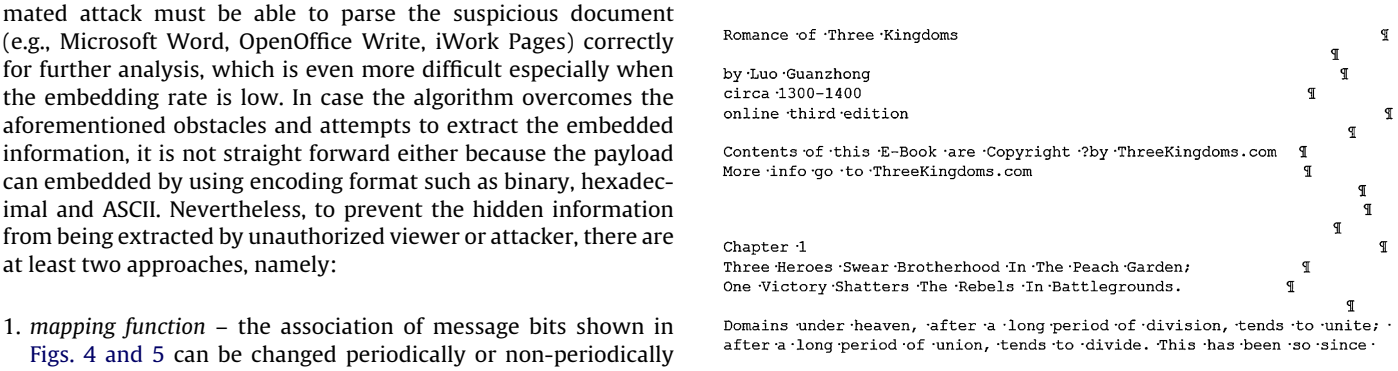

Fig. 7. Output of UniSpaCh after data embedding, with "show/hide formatting marks" feature turned on.

Table 2
Output document file size for various types of cover [kB].

Payload	T_1	T_2	T_3	R_1	R_2	R_3	S_1	S_2	S_3
1	8	8	8	9	8	8	8	8	8
2	13	17	14	15	14	13	14	14	13
4	25	34	25	29	26	25	25	25	26
8	46	58	47	54	50	47	49	49	47
16	89	104	88	96	95	88	95	95	95
32	163	186	164	178	175	162	178	176	174
64	308	342	313	340	327	303	330	333	331
128	611	670	615	663	640	590	642	652	657
256	1198	1292	1197	1310	1269	1171	1270	1299	1324
512	2369	2552	2373	2584	2500	2307	2504	2564	2617
1024	4691	5061	4719	5129	4964	4582	4996	5087	5190

appearance of '·' or '→' in Fig. 7, which suggests imperceptibility of the embedded information. On the other hand, existence of the embedded message is immediately revealed in the case of SNOW, Spacemimic, wbStego4Open and WhiteSteg as shown in Fig. 2. Therefore, we conclude that the proposed UniSpaCh is of higher robustness against DASH when compared to the existing data hiding methods considered.

Next, we consider the embedding efficiency of the existing methods and UniSpach. Random texts of size 1, 2, 4, . . ., 1024 kB are utilized as the payload and embedded into the story *The Romance of Three Kingdom* (Nguyen, 1997) stored in Microsoft Word document format using the existing methods and the proposed UniSpaCh. Here, to clearly evaluate the embedding efficiency of the proposed UniSpaCh, all methods considered are programmed to output just enough cover text (i.e., copied from the input Microsoft Word document, with repetition when necessary) to embed the payload. 1024 kB is set as the upper limit because Spacemimic fails to handle any payload of size greater than 1024 kB. The results are shown in Fig. 8. In general, the output document size of each method increases exponentially as the payload size doubles. Regardless of the size of the payload, UniSpaCh always produces output document of the smallest file size, followed by WhiteSteg, SNOW, Spacemimic and wbStego4Open. Results suggest that, on average, WhiteSteg, SNOW, Spammmic and wbStego4open produce output document of size ∼171%, ∼452%, ∼512% and ∼729% larger than that of UniSpaCh, respectively, when embedding the same amount of information. This trend becomes more obvious when the payload increases to more than 128 kB. In the best case scenario, UniSpaCh produces output document of size almost 9 times smaller than that of the existing method. Similar results are observed when

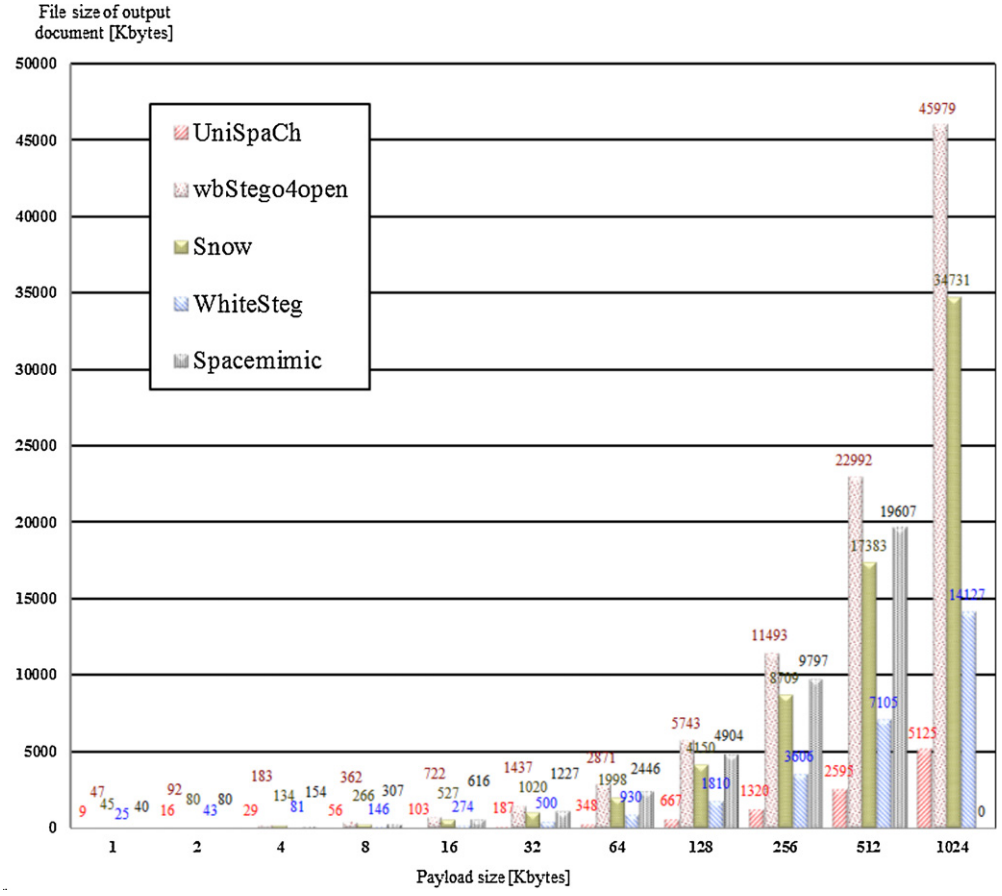

Fig. 8. Results of output document file size vs payload size.

Windows Media Video and JPEG files are utilized as the payload.[1] These results suggest two conclusions; (a) for a given cover document (without copying and appending), UniSpach is able to host more information in terms of raw bit count; and (b) UniSpaCh causes the smallest file size increment when embedding the same amount of information.

Last but not least, we investigate the influence of document content in embedding efficiency for UniSpaCh. Microsoft documents storing thesis (T_i), technical reports (R_i) and stories (S_i) are utilized as the cover for $i \in \{1, 2, 3\}$. Again, random texts of size 1, 2, 4, . . ., 1024 kB are considered as the payload and UniSpaCh outputs just enough cover text to embed the payload. Table 2 records the result for embedding payload of various sizes into different types of document. For any payload size, the results suggest that the output document are of similar file sizes, with fluctuation of ~5.8% on average. Nevertheless, in most cases, R_3 results in the smallest output file size for all sizes of payload. We conclude that the embedding efficiency of UniSpaCh is consistent regardless of content of the host document.

7. Conclusions

A text-based data hiding method called UniSpaCh was proposed to embed payload into Microsoft Word document. A simple attack based on the show/hide formatting mark named DASH was proposed to reveal the existence of embedded information in the conventional methods. Unicode space characters that remained invisible with respect to DASH were utilized together with the ordinary space character to encode payload using inter-word, inter-sentence, end-of-line and paragraph spacings. Results confirmed that UniSpaCh is robust with respect to DASH attack while the existence of externally embedded information was instantly revealed in the case of the conventional methods. UniSpaCh offers higher embedding efficiency when compared to the existing methods. In the best case scenario, UniSpaCh produces output document of size almost 9 times smaller than that of the existing method. Results confirmed that content of the cover document has insignificant influence on embedding efficiency of UniSpaCh.

As future work, we want to further improve the embedding efficiency and imperceptibility of the embedded information without sacrificing carrier capacity. We also want to utilize Unicode characters to hide information in other domains, and investigate the robustness of UniSpaCh against statistical attack targetting on Unicode characters.

References

Allen, J.D., Anderson, D., Becker, J., Cook, R., Davis, M., Edberg, P., 2009. Space characters in the Unicode standard version 5.2 (6.2 general punctuation).

Atallah, M.J., Raskin, V., Crogan, M., Hempelmann, C., Kerschbaum, F., Mohamed, D., Naik, S., 2001. Natural language watermarking: design, analysis, and a proof-of-concept implementation. In: Proceedings of the 4th International Workshop on Information Hiding, pp. 185–199.

Atallah, M.J., Raskin, V., Hempelmann, C., Karahan, M., Sion, R., Topkara, U., Triezenberg, K.E., 2003. Natural language watermarking and tamperproofing. In: Revised Papers from the 5th International Workshop on Information Hiding, pp. 196–212.

Bender, W., Gruhl, D., Morimoto, N., Lu, A., 1996. Techniques for data hiding. IBM System Journal 35, 313–336.

Bergmair, R., Katzenbeisser, S., 2004. Towards human interactive proofs in the text-doma: using the problem of sense-ambiguity for security. In: Zhang, K., Zheng, Y.L. (Eds.), Information Security, Vol. 3225 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 257–267.

Bergmair, R., Katzenbeisser, S., 2007. Content-aware steganography: about lazy prisoners and narrow-minded wardens. In: Camenisch, J., Collberg, C., Johnson, N., Sallee, P. (Eds.), Information Hiding, Vol. 4437 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 109–123.

Bergmair, R., 2004. Towards linguistic steganography: a systematic investigation of approaches, systems, and issues. Master's Thesis. University of Derby.

Calvo, H., Bolshakov, I.A., 2004. Using selectional preferences for extending a synonymous paraphrasing method in steganography. In: Sossa Azuela, J.H. (Ed.), Avances en Ciencias de la Computacion e Ingenieria de Computo-CIC'2004: XIII Congreso Internacional de Computacion. , pp. 231–242.

Chand, V., Orgun, C.O., 2006. Exploiting linguistic features in lexical steganography: design and proof-of-concept implementation. In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences, vol. 6, p. 126b.

Chen, C., Wang, S., Zhang, X., 2006. Information hiding in text using typesetting tools with stego-encoding. In: Proceedings of the 2006 International Conference on Innovative Computing, Information and Control. ICICIC'06, pp. 459–462.

Chotikakamthorn, N., 1998. Electronic document data hiding technique using inter-character space. In: Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems, pp. 419–422.

Culnane, C., Treharne, H., Ho, 2006. A new multi-set modulation technique for increasing hiding capacity of binary watermark for print and scan processes. In: International Workshop on Digital Watermarking, South Korea.

Gutub, A., Fattani, M., 2007. A novel arabic text steganography method using letter points and extensions. In: WASET International Conference on Computer, Information and Systems Science and Engineering (ICCISSE), Viernna, Austria, pp. 28–31.

He, B., Wu, Y., Kang, K., Guo, W., 2009. A robust binary text digital watermarking algorithm for print–scan process. In: World Congress on Computer Science and Information Engineering, vol. 7, pp. 290–294.

Johnson, N.F., Jajodia, S., 1998. Exploring steganography: seeing the unseen. Computer 31, 26–34.

Katzenbeisser, S., Petitcolas, F., 2000. Information Hiding Techniques for Steganography and Digital Watermarking. Artech House Publishers.

Kessler, G.C., 2004. An overview of steganography for the computer forensics examiner. Forensic Science Communications 6.

Khairullah, M., 2009. A novel text steganography system using font color of the invisible characters in Microsoft Word documents. In: ICCEE'09. Second International Conference on Computer and Electrical Engineering, vol. 1, pp. 482–484.

Kwan, M., 2006. The SNOW Home Page. http://www.darkside.com.au/snow/.

Liu, T.-Y., Tsai, W.-H., 2007. A new steganographic method for data hiding in Microsoft Word documents by a change tracking technique. IEEE Transactions on Information Forensics and Security 2 (1), 24–30.

McKellar, D., 2000. Space mimic. http://www.spammimic.com/encodespace.shtml/.

Montalbano, E., 2009. Forrester: Microsoft office in no danger from competitors. http://www.pcworld.com/businesscenter/article/166123/forrester_microsoft_office_in_no_danger_from_competitors.html.

Murphy, B., Vogel, C., 2007. The syntax of concealment: reliable methods for plain text information hiding. In: Processings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents.

Murphy, B., 2001. Syntactic information hiding in plain text. Master's Thesis. Trinity College Dublin.

Nakagawa, H., Sampei, K., Matsumoto, T., Kawaguchi, S., Makino, K., 2001. Text information hiding with preserved meaning – a case for Japanese documents. Information Processing Society of Japan (IPSJ) Transaction 42 (9), 2339–2350, Originally published in Japanese. A similar paper was disseminated by the first author in English and is kept available for download from http://www.r.dl.itc.u-tokyo.ac.jp/ nakagawa/academic-res/finpri02.pdf.

Nguyen, K., 1997. Romance of three kingdoms. http://threekingdoms.com/.

Niimi, M., Minewaki, S., Noda, H., Kawaguchi, E., 2003. A framework of text-based steganography using sd-form semantics model. Information Processing Society of Japan (IPSJ) Transaction 44 (8).

Por, L.Y., Ang, T.F., Delina, B., 2008. Whitesteg: a new scheme in information hiding using text steganography. WSEAS Transaction on Computers 7, 735–745.

Shirali-Shahreza, M., Shirali-Shahreza, M.H., 2007. Text steganography in sms. In: Proceedings of the 2007 International Conference on Convergence Information Technology, ICCIT'07, IEEE Computer Society, Washington, DC, USA, pp. 2260–2265.

Topkara, M., Taskiran, C.M., Delp, E.J., 2005. Natural language watermarking. In: Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents.

Topkara, U., Topkara, M., Grothoff, C., Grothoff, K., 2006. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In: Proceedings of the 8th Workshop on Multimedia and Security, pp. 164–174.

wbStego, 2004. http://wbstego.wbailer.com/.

Westfeld, A., Pfitzmann, A., 1999. attacks on steganographic systems – breaking the steganographic utilities EzStego, Jsteg, Steganos, and S-tools – and some lessons learned. In: Proceedings of the International Workshop on Information Hiding, Dresden, Germany, pp. 61–75.

Wu, M., Liu, B., 2002. Multimedia Data Hiding. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Xuan, G., Shi, Y.Q., Ni, Z., Chai, P., Cui, X., Tong, X., 2007. Reversible data hiding for jpeg images based on histogram pairs. In: ICIAR, pp. 715–727.

Yu, L., Niu, X., Sun, S., 2005. Print-and-scan model and the watermarking countermeasure. Image and Vision Computing 23, 807–814.

Zou, D., Shi, Y.Q., 2005. Formatted text document data hiding robust to printing, copying and scanning. In: ISCAS, vol. 5, pp. 4971–4974.

[1] SNOW and Spacemimic can only embed text file while UniSpaCh can handle all types of file as payload.

Lip Yee Por received his PhD, B. Comp. Sc. and M. Comp. Sc. at University of Malaya, Malaysia. He is a lecturer in the Faculty of Computer Science and Information Technology at University of Malaya since 2004. His research areas include information hiding, steganography, graphical authentication and grid computing. He is a Senior Member of IEEE and his biography has been included in Marquis Who's Who in the World.

KokSheik Wong received the B.S. and M.S. degrees in both computer science and mathematics from Utah State University, USA, in 2002 and 2005, respectively. In 2009, he received the Doctor of Engineering degree from Shinshu University, Japan, under the scholarship of Monbukagakusho. In 2010, he joined the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia, where he is currently a senior lecturer. His research interests include information hiding, steganography, watermarking, multimedia content scrambling, multimedia signal processing, and their applications. Dr.Wong is a member of IEEE.

Kok Onn Chee received his B. Comp. Sc. at University of Malaya, Malaysia. He is currently pursuing his Master of Computer Science at University of Malaya, Malaysia.