A Hybrid Natural Language Information Hiding System

Lu He[1,2], Xiaolin Gui[1], Reifeng Wu[2], Biqing Xie[2], Chang Hu[3]
[1]*School of Electrical and Information Engineering,*
Xi'an Jiaotong University, Xi'an, 710127, China
[2]*School of Information Science and Technology,*
Northwest University, Xi'an 710127, China,
[3] *Xi'an Institute of Measurement Technology,*
Xi'an 710068, China, phone: 86-29-88308273
xlgui@xjtu.edu.cn

***Abstract*—Natural language information hiding technique is a hotspot in information security field in recent years. However, the extant algorithms still face some serious problems, such as insufficient capacity, nonuniform distribution of cover unit, and lack of studies on data-hiding codes. In this paper, to get over above problems we propose the concept of cover unit which erase the differences between the natural language processing techniques. So, we can use multiple natural language processing techniques at the same time. According to this approach, we propose general embedding/extracting algorithms and develop a hybrid natural language information hiding (HYNLIH) system. The experiment results show HYNLIH achieve higher capacity, cover units distribute more uniformly, security and robustness are also improved when steganalysis and attacks are presented.**

***Index Terms*—Cover unit, data-hiding code, natural language information hiding, natural language processing.**

I. INTRODUCTION

Information hiding studies methods to make private messages are embedded in seemingly innocuous cover messages [1]. The main sub-disciplines of information hiding are steganography, which is about concealing of the content of secret message very existence for covert communication, and watermarking, which is about adding invisible attribution data to media files for verify its authenticity, ownership, copy control or annotation data. The to-be hidden message is called secret message or secret bits. Natural language information hiding is the art of using written natural language to conceal secret messages by making meaning-preserving transforms to plain text that can pass human and machine detection [2]. The aims of natural language information hiding are similar to those in multimedia techniques. The cover texts are not only composed of natural language texts, but also are generated to have a cohesive linguistic structure. However, the generating methods are not suitable for watermarking. Thus, in this paper, we ignore the generating methods and propose a system which is as general as possible to avoid focusing on steganography scenarios only, so as to encompass as many natural language information hiding applications as possible.

Natural language information hiding is depending on synonymy, movement of phrasal constituents, syntactic or semantic transformation techniques. We call it cover unit which is a segment of a text that can be transformed with meaning preserving. In contrast to rich media files such as audio and images, it has been proven difficult to embed hidden secret bits in plain text files because the natural language processing (NLP) techniques are not yet accurate and robust [3]–[9]. Generally speaking, the natural language information hiding mainly suffered the following problems: 1) lack of embedding room. Size of text, such as news, is very small, for example, only hundreds bytes or hundreds words, while size of image is usually several hundred KB or tens of thousands pixels; 2) Cover units distribute nonuniformly in text, which degrades security and robustness. 3) The security and robustness of information hiding mainly depends on data-hiding codes [10]–[13]. However, it brings very different background knowledge together: natural language processing (NLP) techniques and electrical engineering. Most natural language data hiding researchers come from NLP areas and they are unfamiliar with the data-hiding codes, while researchers coming from electrical engineering are proficient at signal processing and coding techniques but it is difficult for them to understand the specificity of the natural language processing techniques.

In this paper, we review existing natural language information hiding schemes, and point out some challenges that natural language information hiding should overcome. We propose general embedding and extracting algorithms which combine multiple NLP tools to overcome these challenges. For validating our algorithms, we implement the Hybrid Natural Language Information Hiding (HYNLIH) System which separate NLP techniques from coding very well. Expiraments results show HYNLIH achieved higher performance in capacity, robustness, and security.

II. THE STATE–OF-ART AND CHALLENGES

While natural language information hiding and multimedia

Manuscript received March 19, 2012; accepted May 15, 2012.
This paper is supported by the National Natural Science Foundation of China (No. 60873071, 61172090), Science Research Plan of Shaanxi Education Ministry of China (12JK0742), Research Plan of Shaanxi Technical quality Ministry of China (2010-17, 2010-20).

information hiding share common goals, they employ very different technologies. So far natural language information hiding is depending on synonymy, movement of phrasal constituents, and semantic transformation techniques.

A. The synonym substitution method

Winstein [3] firstly brought forward a lexical steganography system, T-Lex, which is based on synonym substitution. T-Lex has a synonym dictionary in which the synonyms are extracted from WordNet [4]. In order to ensure that only words with close senses are replaced with each other, only such words completely in the same synsets are grouped in the same synonym set. Such synonym set is called absolute synonym by Bolshakov [5]. Synonyms in the same synonym set are numbered from 0. Thus, every synonym in a cover text is a digit that may belong to different radix. Assuming there are N absolute synonym words in a cover text, so the N numbers can be extracted from the cover text and viewed as an N-digit mixed-radix number. The secret message can be treated as a number. So, the embedding processing makes the mixed-radix number presented by a cover text equal to the secret number by synonym substitutions and the extracting process reads the mixed-radix number presented by the stego-text.

In order to increase capacity, Bolshakov gave the definition of relative synonym that words in the same synonym set are synonyms in some contexts, but are not synonyms in other contexts [5]. Avoiding the improper synonym substitution, words can be substituted with relative synonyms only if the latter form valid collocations with the context according to the statistics gathered from Internet [5].

The shortcoming of such synonym substitution methods is that they do not agree with the genre and the author style of the given text. Taskiran et al. [6] used a universal steganalysis method based on language models and support vector machines to differentiate sentences modified by a lexical steganography algorithm from unmodified sentences. However, Taskiran's method needs a lot of innocuous texts of the same author for getting an author's style is not feasible. Even getting the author's style, the trigram models are not accurate enough so that the false alarm is too high. Luo et al. [7] found that the synonym substitution led to the phenomenon that the probability of synonym pair presentation in the cover text increases. In the light of this observation, the author proposes a steganalysis algorithm utilizing the number of synonym pair presentation to decide whether the hidden message exists in text or not. Experimental results show that the accuracy achieves 86.2%.

B. The syntactic transformation method

Early syntactic transformations depended on deep structure analysis technologies. Atallah et al. [8] described a proof-of-concept watermarking implementation based on sentence transform by which the meaning can be preserved. The selected sentences carrying the secret bits information depends only on the tree structure and proceeds as follows: The nodes of the tree T_i for sentence s_i of text are labeled in pre-order traversal of T_i. Then, a node label j is converted to 1 if $j + H(p)$ is a quadratic residue modulo p, and to 0 otherwise, where p is a secret key and $H(\cdot)$ is a one-way hash function. A node label sequence, B_i, is then generated by traversing T_i according to post-order. A rank, d_i, is then derived for each sentence for s_i using $d_i = H(B_i)$ XOR $H(p)$ and the sentences are sorted by rank. Starting from the least-ranked sentence s_j, the watermark is inserted to s_j's successor in the text. The bits are stored by applying syntactic transformations, such as Adjunct Movement, Passivization.

Some researchers try to avoid doing deep structure parse. Murphy and Vogel [9] presented three natural language marking strategies based on fast and reliable shallow parsing technologies, which relies on part-of-speech (POS) tagging, and on widely available lexical resources: lexical substitution (or absolute synonym), adjective conjunction swaps, and relativiser switching.

Ma et al. [10] pointed out the number of sentences that can be transformed are few and can be identified easily. They proposed an attack scheme: firstly, choose some of sentences that can be transformed. Then, these sentences are automatically transformed by the same method as the embedder used. Experiment shows few of transformation can destroy the watermarks efficiently.

C. The semantic transformation method

The method used in [11] for generating meaning-preserving semantic transformations is mainly depending on the usage of noun phrase coreferences. Two noun phrases are coreferent if they refer to the same entity. Based on the coreference concept, different transformations may be introduced. One is co-referential pruning, where repeated information about the coreferences is deleted. The opposite side of this operation, coreferent grafting, may also be performed while information about a coreference is repeated in another sentence, or added to the text using a fact database. The method embed secret message in the tree structure is same as [8]. Difference between the two algorithms is that the first one modifies syntactic parse tree of the cover text sentences while the second one modifies the semantic tree.

However, the current NLP techniques could not offered adequate tools yet for semantic parsing, which would lead to semantic watermarking [12]. This method class is proof-of-concept by assuming a perfect parser or verified by hand on corpus that are parsed with syntactic or semantic trees.

D. The challenges

In contrast to rich media files such as audio and images, it has been proven difficult to embed hidden secret bits in plain text files [9]. Generally speaking, the natural language information hiding mainly suffered the following problems:

Lack of embedding room. For example, according to our statistics, the capacity of T-Lex is about 2.02 absolutely synonyms per 100 words. Although the capacity increased prominently by using relative synonym, the accurate word sense disambiguate tools are still struggle to achieve much more than 60% accuracy on general text. Thus, much relative synonyms have to be given up [9]. To syntactic transformation techniques, there are about 6-8 transformable sentences per 100 sentences in English [10]. The best capacity is Meral's method for Turkish, averagely 0.81 bit can be

embedded within a sentence [12], but his method is only suitable for agglutinative languages. Consequently, a quite long text is need for hiding a short message, not to mention considering security or robust encoding that would increase the length of the to-be hidden bits further.

The non-uniform of cover units in texts. For example, although capacity of Meral's method is achieved on the average 0.81 bit per sentence by 20 natural language processing tools, there is about 25% sentences in text cannot be performed by any transformation [12]. Ma makes use of the nonuniform distribution phenomenon to narrow the attack targets [10], and our experiments shows that with more uniform distribution, security and robustness will be improved.

The security and robustness of information hiding are largely depending on data-hiding codes [13]. However, most researchers are putting their focus on how to perform meaning-preserve transformations. The reason why they neglect the data-hiding codes is their unfamiliarity with signal processing and coding theory. Whereas researchers who are proficient at signal processing and coding techniques are hard to understand the specificity of natural language. It is urgent to comb advantage knowledge of researchers with very different backgrounds in order to avoid the unfamiliar area.

The first problem is hard to be solved unless NLP techniques get breakthrough. As for the second problem, it is impossible to settle if only one NLP technique is adopted. As for the third problem, separated hide-data coding algorithm from embedding and extracting processes are demanded.

III. General embedding and extracting algorithms

We noticed that, no matter synonyms, syntactical structure of sentence or other ways holding the secret bits are all minimal segments of text that can be processed by a given NLP tool for meaning-preserve transformation. Hence, we introduce the following definitions:

Definition 1: Given a NLP tool, a cover unit is a minimal segment in a cover text that can be processed by the NLP tool for meaning-preserve transformation.

Definition 2: An equivalent transformation is a word or words that convey the same, or almost the same meaning of a cover unit.

Definition 3: A substitution set is composed by a cover unit and all of its equivalent transformations. That is to say, element in a substitution set can exchange each other in a given context.

Despite different NLP tools are used for embedding or extracting, by introducing the concept of cover unit the hidden message can be seen as substituting a word or words with other word or words conveying the same, or almost the same meaning, which indicate that the data-hiding code algorithm is independent from the NLP tools. Furthermore, whatever the NLP tools are used, the cover units are homogeneous. Thus, any NLP technique can be combined together to increase the capacity and improve the uniformity of cover unit distribution.

According to this method, we re-planned the embedding and extracting processing, and propose general embedding and extracting algorithms as following:

Let $P = y$-sentences text $\{p_1, p_2, \ldots, p_y\}$. $C = \{c_1, c_2, \ldots, c_i\}$ is a data-hiding code plug-in set, c_i is the i-th data-hiding code plug-in. $M = \{m_1, m_2, \ldots, m_j\}$ is a cover manipulation plug-in set, m_j is the j-th cover manipulation plug-in.

The Embedding Algorithm:

1) Load $c_k(1 \leqslant k \leqslant i)$ and $\forall m \in M$;
2) Foreach $p_l(1 \leqslant l \leqslant y)$
   a) Foreach $m(m \in M)$
      i. Parse p_l by m to get cover units and generate equivalent transformations for every cover unit.
      ii. For each cover unit, the bits strings were assigned to each equivalent transformation by c_k.
3) Sort and block the cover units by c_k, the order before sorting will be saved in *seq* and the order after sorting will be saved in *sn*;
4) Encode the secret bits into data-hiding code according to the blocking result by c_k;
5) Sort the data-hiding code bits by c_k according to the mapping between *seq* and *sn*;
6) foreach $p_l(1 \leqslant l \leqslant y)$
   a) foreach $m(m \in M)$
      i. Choose right equivalent transformations to substitute cover unit to generate stego-text by m according to the sorting data-hiding code bit;
7) Return the stego-text;

The Extracting Algorithm:

1) Load $c_k(1 \leqslant k \leqslant i)$ and $\forall m \in M$;
2) Foreach $p_l(1 \leqslant l \leqslant y)$
   a) foreach $m(m \in M)$
      i. Parse p_l by m to get cover units and generate equivalent transformations for each cover unit.
      ii. For each cover unit, the bit strings were assigned to each equivalent transformation by c_k.
3) Sort and block the cover units by c_k, the order before sorting will be saved in *seq* and the order after sorting will be saved in *sn*;
4) Read the confusing data-hiding code bits from the cover units by c_k.
5) Decode the data-hiding code bits to recover the secret bits by c_k.

The embedding algorithm can be divided into three stages. In the first stage, there are three things should be done. Firstly, if multiple cover manipulation plug-ins were applied, they must be followed into a certain order, such as alphabetical order, for correct embedding and extracting.

Secondly, every cover manipulation plug-in utilizes a NLP tool to parse the cover text. This would help to find the cover units out and generate equivalent transformations. Then we could compose substitution sets for every cover unit. For example, an English absolute synonym plug-in would compare each word of a cover text to the synonym dictionary of T-Lex. As an entry of the synonym dictionary, a word can be deemed as a cover unit and its synonyms are equivalent transformations. Another example is English syntactic plug-in.

The plug-in could utilize a syntactic parser, in our experiment the Stanford Parser [14] is used, to get syntactical structure of each sentence in a cover text. If the syntactical structure of one sentence can be applied to syntactic transformations, the sentence is a cover unit. Some of the common syntactic transformations in English were listed in [8].

Thirdly, since the hidden message is embedded by substituting a word or words with equivalent transformation, every equivalent transformation of the same substitution set should be assigned into a bit or several bits. Suppose u_i is a cover unit and s_i is the substitution set of u_i. The data-hiding code plug-in assigns a bit or several bits to each equivalent transformation of s_i under control of the secret key. For example, if s_i has only two equivalent transformations, the u_i could be used for embedding only one bit at most; if s_i has forth-equivalent transformations, the u_i could be used for embedding one bit, i.e. two transformations are assigned 0 and the other two equivalent transformations are assigned 1. Otherwise, the transformations would be embedded into two bits, for each of the equivalent transformation is assigned to different bits string.

At the second stage, data-hiding code plug-in translates secret bits into data-hiding code. The following things should be done:

Some data-hiding code algorithms need to examine the cover features during the embedding processing, such as informed embedding schemes [15]. Since every cover unit can be viewed as a feature, all cover units of the cover text would compose a feature set of the cover text. Informed embedding scheme examine these cover units before encoding. That is why the embedding algorithm should be find out all cover unit first.

Secondly, if we embed the secret bits in cover units orderly, the adversary could read them out easily. Thus for keeping secret, a data-hiding code plug-in should sort cover units out under the control of the secret key. Similar to cryptography, many data-hiding codes are block codes, which group the cover units into blocks. For example, F5 code [16] would group a sequence of secret bits with length n into a block, then encodes the block of secret bits into an F5 code with length 2^n-1 bits. If a cover unit holds one bit, the 2^n-1 cover units are composed one block for holding the n secret bits. [8] embeds β bits into one sentence that can be performed meaning-preserved transformation, and a block shall be composed one sentence.

Furthermore, the cover manipulation plug-in should substitute the cover units with one of its equivalent transformation according to the encoding result. However, since the cover units were sorted by the data-hiding code plug-in, in order to embed a bit, the cover manipulation plug-in must search the cover unit from beginning of the cover text. Performance of such strategy is not good. In order to avoid multi-scan the cover text, the encoding result should be sorted as the same order of cover units. For example, suppose the cover units list is {a, b, c, d, e, f} before sorting the units cover and {c, e, d, a, f, b} after sorting. Suppose two cover units composed one block, and the blocking result is {(c, e), (d, a), (f, b)}. The mapping between cover units and encoding bits is shown in Table I.

TABLE I. EXAMPLE OF SORTING COVER UNITS AND ENCODING BITS.

Cover unit list before sorting	Cover unit list after sorting	The encoding result	The sorting bits
a	c	1	1
b	e	0	1
c	d	1	1
d	a	1	1
e	f	0	0
f	b	1	0

The last stage in embedding algorithm is generating a stego-text. The cover manipulation plug-in would read the cover unit list according to the order before sorting, and scan the cover text to find the current cover unit out. Following, the cover manipulation plug-in will choose equivalent transformation according to the sorting encoding result and substitute the cover unit for embedding. Thus, for generating a stego-text, we only need a one-pass scan cover text.

In the extract processing, the cover manipulation plug-ins used in the embedding process would parse the stego-text for finding the cover units out and generating equivalent transformations, just like the embedding processing does. Then the data-hiding code plug-in would assign bits string to each equivalent transformation, and read the encoding bits to sort and block just like what happened during the embedding processing. Lastly, the data-hiding code plug-in would decode for recovering the secret messages.

IV. EXPERIMENTS

In order to validating our algorithms, we developed the HYNLIH system according to the general embedding and extracting algorithms, which is implement in three-tier architecture. The topmost tier would be the presentation tier which provides the human-computer interface. The general embedding algorithm and extracting algorithm implement by the middle tier. The lowermost is the cover manipulation plug-ins and data-hiding code plug-ins which are loaded by middle tier.

We designed five schemes for contrasting capacity, security and robustness of the NYNLIH. Scheme No.1 used the absolutely synonym substitution cover manipulation plug-in as [3]. Scheme No. 2 used the relative synonym substitution cover manipulation plug-in as [5]. Scheme No. 3 used the syntactic transformation cover manipulation plug-in as [8]. Scheme No. 4 used substituting the swapping of complementisers and relativisers cover manipulation plug-in as [9]. Schemes No. 1-4 are classical natural language information hiding schemes. Scheme No.5 combined all cover manipulation plug-ins that are used in scheme No.1-4. Scheme No.1-5 are used the same data-hiding code plug-in, that is random codes plug-in. The random code plug-in sorts cover units randomly so that the adversary cannot indicate where the secret bits are embedded. The performance of random code plug-in is similar to quadratic residues coding (which is a classical encoding scheme suitable for syntactic and semantic natural language information hiding) but independent of syntactic or semantic tree and hence suitable to various cover manipulation plug-in. Then, we collected a

large of raw text materials from the Internet, which involve English Classic, news, economy, health, sports, and technology etc. We chose 1000 texts as innocent text for learning the innocent pattern by steganalysis algorithm and the other 1000 texts as cover text for embedding secret message. Lastly, we embedded secret message in cover texts according by the five schemes respectively and got 5000 stego-text. We do not estimate the imperceptibility because scheme No.1-4 are simulated the existing algorithms.

A. The Capacity and Cover Unit Density

In order to measure the capacity in each scheme, we introduced the definition of cover unit density as following

$$U = C/N, \tag{1}$$

where N is the number of words in a cover text, C is the number of cover units that cover manipulation plug-ins may find in this cover text. We plotted the cover unit density of every cover text against the five schemes in Fig. 1 (a) – (e). The graphs clearly show that the highest density of cover unit is in scheme No.5, which means the distribution of cover unit in scheme No.5 is the most uniform one while scheme No.3 is the most dispersive scheme, which means the distribution of cover unit in scheme No.3 is the most non-uniform one. The densities of cover unit of top 100 cover texts are plotted in Fig. 2 for showing details. The horizontal dashed lines in Fig. 2 stand for average capacity of 1000 cover texts. Obviously, the capacity of scheme No.5 is equal to the sum of the capacity of scheme No.1–4.

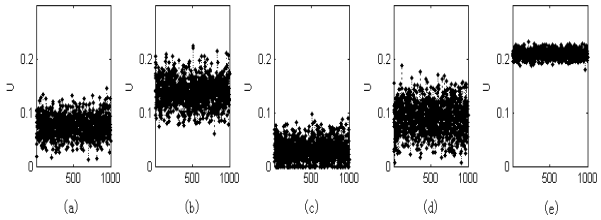

Fig. 1. Cover unit densities of the 1000 cover texts. The x- axis is the text id and the y-axis is the cover unit density.

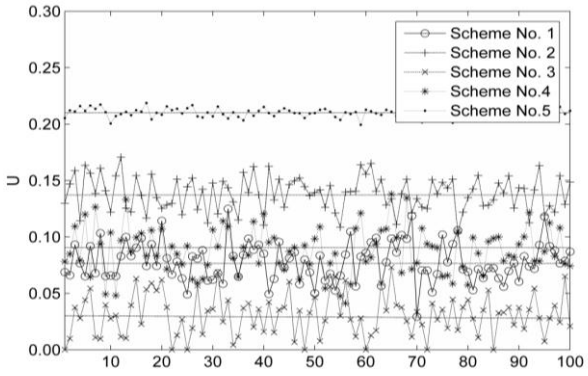

Fig. 2. The details of cover unit densities of top 100 cover text. The horizontal dashed lines are the average capacity.

B. The Security

We performed the steganalysis algorithm proposed in [7] on scheme No.1, No. 2 and no.5. The results are shown in Fig. 3. As the results shown, obviously, the difference between the features of innocent text and the features of hybrid embedded stego-text is smaller than the difference between the features of clean text and the features of single embedded stego-text. The superior performance of scheme No. 5 is because the steganalysis is based on detecting differences of the cover texts before and after embedding. Each steganography method embeds in texts in the same way and produces a particular type of distortion on stego-text. Therefore, discovering the distortion type of a steganography method, namely the difference of some statistical characteristics between the innocent texts and the stego-texts is the key issue in steganalysis. The accuracy of steganalysis is proportionate to the embedding rate, since the more secret bits embedded into a cover text the more difference between the innocent texts and the stego-texts. The HYNLIH employs multiple cover manipulation plug-ins, it reduces the embedding rate on one cover manipulation tool, that is to say, many synonyms are not used when embedding. Thus, the number of synonym pair is reduced that the different is nondistinct between the curve C_D/N_D (C_D/N_D is the proportion of the number of synonym pair in stego-texts and in innocent texts) of innocent texts and stego-texts.

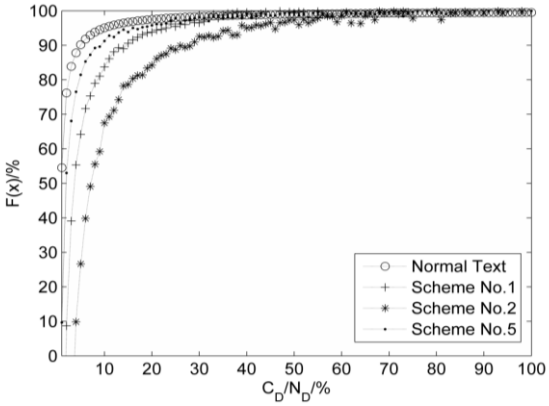

Fig. 3. The distribution curve of C_D/N_D. The x-axis is the number of synonym pair rate (denote by C_D/N_D). The y-axis is the cumulative distribution function of C_D/N_D.

C. The Robustness

We performed Ma's attack [10] against scheme No.3-5. The result is shown in Fig. 4.

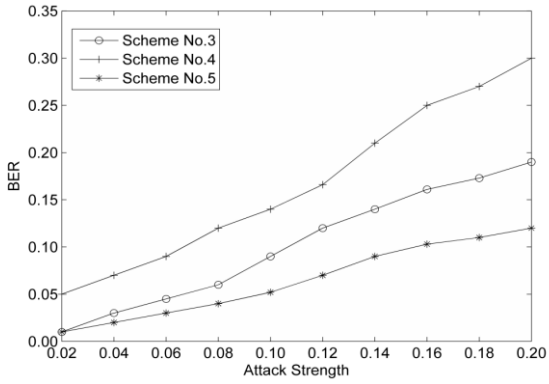

Fig. 4. The bit error rate (BER) of syntactic transformation attack. The x-axis is the percentage of words that affected by sentence transformation. The y-axis is the bit error rate.

The reasons of superior performance of scheme No.5 are similar to the analysis in section 4.2, that is to say, since only a part of secret bits are embedded into syntactic structure, the

sentences attacked may not carry bits. Thus, the error bit rate is reduced.

V. CONCLUSIONS

In this paper, we proposed the concept of cover unit for combining multiple natural language process tools and overcame some challenges in natural language information hiding. We divided the natural language information-hiding algorithm into two groups: cover manipulation plug-in and data-hiding code plug-in. And the general embedding algorithm and extracting algorithm were proposed which may combine the cover manipulation plug-ins together for data-hiding coding. Then we implemented a Hybrid Natural Language Information Hiding system, the HYNLIH. Experiments show the HYNLIH achieved a higher capacity, higher security, higher robustness, and the cover units distribute more uniformly. In the future, we will try to integrate more cover manipulation plug-ins and data-hiding code plug-ins to validate our general embedding and extracting algorithm, and measure the capacity.

REFERENCES

[1] S. Katzenbeisser, F. A. P. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House Inc., 2000, pp. 21.

[1] M. Topkara, C. M. Taskiran, "Natural language watermarking", in *Proc. of the SPIE*, vol. 5681, SPIE press, 2005, pp. 441–452. [Online]. Available: http://dx.doi.org/10.1117/12.593790

[2] K. Winstein, *T-Lex*. [Online]. Available: http://alumni.imsa.edu/~keithw/tlex

[3] *WordNet: An Electronic Lexical Database*, MIT Press, 1998, pp.32.

[4] I. A. Bolshakov, "A method of linguistic steganography based on collocationally-verified synonymy", in *Proc. of the 7th Information Hiding*, Springer, 2005, pp. 607–614.

[5] C. Taskiran, U. Topkara, M. Topkara, et al., "Attacks on Lexical Natural Language Steganography Systems", *in Proc. MM&Sec '06*, vol. 6072, SPIE press, 2006, pp. 97–105,.

[6] G. Luo, X. M. Sun, L. Y. Xiang, et al., "Steganalysis on Synonym Substitution Steganography", *Journal of Computer Research and Development*, vol. 10, no. 45, pp. 1696–703, 2008.

[2] M. J. Atallah, V. Raskin, M. Crogan, et al., "Natural Language Watermarking: Design, Analysis, and a Proof- of- Concept Implementation", in *Proc. of the 4th Information Hiding*, Springer, 2001, pp. 185–200. [Online]. Available: http://dx.doi.org/10.1007/3-540-45496-9_14

[7] B. Murphy, C. Vogel, "Statistically Constrained Shallow Text Marking: Techniques, Evaluation Paradigm, and Results", in *Proc. of the SPIE*, SPIE press, vol. 6072, 2006, pp. 1–12.

[8] G. P. Ma, Y. X. Zhang, L. He, et al., "Active Attacks on Syntactic Natural Language Steganography", in *Proc. of the 9-th China Information Hiding*, Sichuang University press, 2010, pp. 54–58.

[9] M. J. Atallah, V. Raskin, C. F. Hempelmann, et al., "Natural Language Watermarking and Tamperproofing", in *Proc. of the 5th Information Hiding*, Springer, 2002, pp. 196–212.

[3] H. M. Meral, B. Sankur, A. S. Ozsoy, et al., "Natural language watermarking via morphosyntactic alterations", *Computer Speech and Language*, Elsevier Science, vol. 1, no. 23, pp. 107–125, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.csl.2008.04.001

[10] P. Moulin, R. Koetter, "Data-Hiding Codes", IEEE press, vol. 12, no. 93, 2005, pp. 2083–2126.

[11] *The Stanford Parser*. [Online]. Available: http://nlp.stanford.edu/software/lex-parser.shtml

[12] M. Barni, F. Bartolini, *Watermarking Systems Engineering: Enabling Digital Assets Security and Other Application*. Marcel Dekker Inc., New York, USA, 2004, pp. 45–47.

[4] A. Westfeld, "F5 a steganographic algorithm: high capacity despite better steganalysis", in *Proc. of the 4th Information Hiding*, Springer, 2001, pp. 289–302. [Online]. Available: http://dx.doi.org/10.1007/3-540-45496-9_21