Multimed Tools Appl (2018) 77:28969–28989
https://doi.org/10.1007/s11042-018-6072-8

A linguistic steganography based on word indexing compression and candidate selection

Lingyun Xiang[1,2] · Wenshuai Wu[2] · Xu Li[3] · Chunfang Yang[4]

Received: 11 September 2017 / Revised: 7 April 2018 / Accepted: 29 April 2018 /
Published online: 11 May 2018

Abstract In this paper, a novel linguistic steganography with high imperceptibility and undetectability is proposed via secret message compression and candidate text selection. The length of the practical embedded payload can be reduced by the proposed word indexing compression algorithm(WIC), while a best stego text with high undetectability can be selected from candidates by the stego text selection strategy. WIC algorithm losslessly compresses the secret message by combining a minimum maximum weight algorithm with Huffman coding under the help of the candidate cover text. To improve the anti-steganalysis capability, ten cover texts with small compression ratios are selected from a huge cover text set, and are embedded the corresponding compressed secret message by using synonym substitutions. Only one stego text is selected by a given rule derived from the distance between a cover text and its stego text. Experimental results show that the proposed compression algorithm achieves better compression ratios than Huffman and LZW coding algorithms leading to higher embedding efficiency, and our steganography performs well in anti-steganalysis capability with compression and the stego text selection rule.

Keywords Linguistic steganography · Steganalysis · Compression · Selection · Synonym substitution · Word indexing

✉ Lingyun Xiang
xiangly210@163.com

[1] Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, Changsha University of Science and Technology, Changsha, 410114, China

[2] School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, 410114, China

[3] School of Maths and Computing Science, Changsha University of Science and Technology, Changsha, 410114, China

[4] Zhengzhou Science and Technology Institute, Zhengzhou, China

1 Introduction

Nowadays, it is very important of protecting digital information against illegal attacks in the network. There are many security techniques to achieve secure communication, such as the encryption [12, 22], the watermarking [13], the steganography [8], and so on. Steganography is the art of embedding secret message into the cover objects, such as images, videos, texts, speeches in a covert way, aiming to transmit stego objects with secret messages unsuspectedly.

Among all of the aforementioned cover objects, text data is the most common and excellent cover object for steganography, as its great prevalence in real life. The native statistical properties of text data have been widely employed by existing steganographic methods [2], which adopt diverse linguistic transformations, such as lexical substitution [4], syntactic transformation [15], and semantic transformation [1] to modify text data, making modifications imperceptible to the observer.

Synonym substitution is one of the common linguistic transformations, and it utilizes some similar or relative words named synonym for text steganography. Winstein [19] firstly distributed a practical linguistic steganographic system Tyrannosaurus Lex(abbreviated as T-Lex) online, where a block encoding method encoded absolute synonyms from the electronic dictionary WordNet, and a synonym substitution mechanism embedded the secret message bits. To avoid decoding ambiguities, each synonym was encoded as a unique codeword for steganography. Similar to T-Lex, Muhammad et al. [16] adopted two selected absolute synonymous words from each synonym set for pursuing good imperceptibility. However, many other relative synonyms are discarded, which leads to low embedding capacity.

For enhancing the embedding capacity and selecting suitable synonyms, Bolshakov [4] applied transitive closure to merge all the overlapping synsets into one set, and used the collocation-test to remove unsuitable words from the merged synsets, making each synonym belong to the same synonym set. Similar to Bolshakov's method, Chang et al. [5] treated words as vertices in a graph, and linked synonyms by edges. A vertex coding algorithm was developed to encode each synonymous word to a unique bit string without discarding any synonyms. Besides, Google n-gram corpus was used for checking the applicability of each synonym to ensure information security. To improve the acceptability of synonyms in context, Liu et al. [14] gave a disambiguation function to select correct synonyms to substitute original words. Topkara et al. [17] proposed a quantitative resilience criterion to prioritize multiple alternatives and selected the more ambiguous words from synonym sets.

The aforementioned methods improved the quality of the stego text and obtained good imperceptibility. However, some changes are made on statistical features of the cover text inevitably, making embedded secret messages in high risks of being detected by linguistic steganalysis [6, 20]. In order to ensure the security, secret messages must be embedded without significant distortion on the cover objects. Hu et al. [11] gave a feasible steganography to preserve the word frequency distribution of the cover text unchanged. However, only when synonymous words appear in a cover text for more than one time, they can be used to carry secret information, which leads to low embedding capacity. In image steganography, matrix embedding [18] introduced by Crandall [7] is the most well-known approach to reduce the distortion measured by the number of embedding modifications. Matrix embedding was proved to be efficient for large payloads which uses of the codes constructed from simplex codes and random linear codes of small dimension [9]. Therefore, Yang et al. [21] attempted to apply the matrix embedding into the synonym-substitution

based steganography to increase its anti-steganalysis capability. However, this method relies on embedding capacity to improve security, which deteriorates the practicality of linguistic steganography, whose embedding capacity is usually very low. Besides, the reduction of embedding modifications cannot protect stego objects from being discovered by all kinds of steganalysis.

A good linguistic steganography with high imperceptibility should achieve high embedding capacity and anti-steganalysis capability. However, no existing methods could solve this problem in a proper way. In this paper, we propose a new linguistic steganography to improve both the embedding capacity and anti-steganalysis capability. The main contributions include two following parts.

1) The wording indexing compression, which builds an index table for the words in the secret message by finding their positions in each cover text, is proposed to improve the embedding capacity. In addition, a minimum maximum weight algorithm is employed to find an optimal path, which has shortest compression length for retrieved secret words. Combining with Huffman coding for the unretrieved secret words, the secret message can be effectively compressed. With the compressed secret message, the length of the practical embedded payload is reduced, thus the relative embedding capacity is greatly increased, as much more original secret messages can be embedded. On the other hand, the modifications made into the cover text will be effectively reduced improving the embedding efficiency;

2)The stego text selection strategy is designed properly to select a more undetectable stego text achieving high anti-steganalysis capability. In details, ten cover texts with relative low compression ratios will be selected and embedded the compressed secret message. In the end, only one stego text with shortest distance from the corresponding cover text is reserved. The difference between the stego text and the cover text is imperceptible, and the improvement of the capability of anti-steganalysis is significant.

The rest of this paper is organized as follows. In Section 2, the framework of the proposed steganography is described. In Section 3, the secret message compression based on word indexing is presented. Section 4 introduces the stego text selection process. Section 5 details the experimental results and analysis. Finally, the conclusions are given in Section 6.

2 The proposed linguistic steganography framework

Size of the message that can be embedded is limited to the number of synonyms contained in the cover text, which is always small. Thus, it is difficult for the existing linguistic steganographic methods based on synonym substitution providing considerable embedding capacity. Low embedding capacity will make that a linguistic steganography is insignificant to practical application. However, some improvements of the embedding capacity are inevitably compromise the security. Steganography with low security cannot prevent secret messages from being detected by steganalysis efficiently.

In order to improve the embedding capacity and statistical security at the same time, we propose a novel linguistic steganography by using compression and selection strategy. The framework of the proposed steganography is shown in Fig. 1. It mainly has three processes: the secret message compression, the stego text generation and the secret message extraction. The secret message compression uses a word indexing compression algorithm to improve embedding capacity by reducing the practical embedding operations. The stego text generation designs a stego text selection strategy to enhance the capability of anti-steganalysis

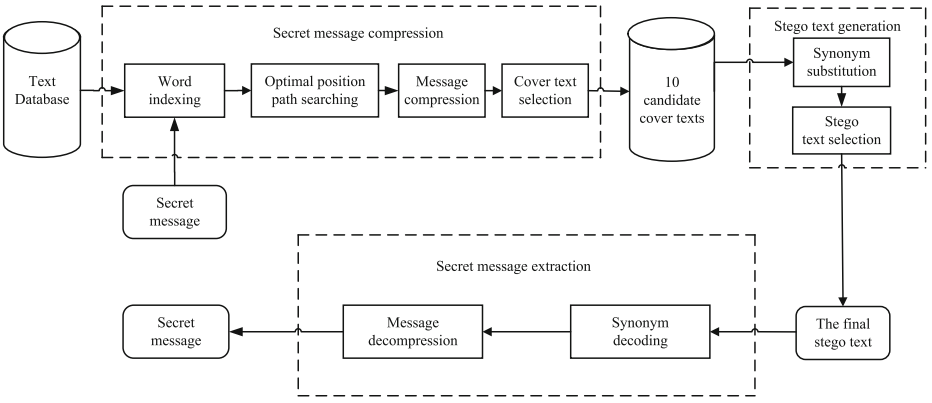


Fig. 1 The framework of the proposed linguistic steganography

by selecting a good stego text with small distortion. The secret message extraction correctly recovers the secret message from the selected stego text.

(1) Secret message compression

The secret message compression losslessly compresses the secret message into an extremely short string by combining a text database, which is firstly built by collecting the huge texts from the Internet. Each text is regarded as a retrieved text. By the secret message compression, a shorter message will be embedded. In this paper, each retrieved text and secret messages all in English. And the secret message, which is necessary to be constrained as a natural language text, is segmented into a token sequence for compression. The tokens mainly include the words and the punctuations, which in this paper are all called secret words.

In order to compress the secret message efficiently, an index table of the secret words is built for each retrieved text in the database, and the secret message is then converted into a compressed binary sequence by the following steps.

Step 1: Word indexing. The purpose is to construct a word index table to record the positions of each secret word within a retrieved text.

Step 2: Optimal position path searching. The secret words in the retrieved text are called retrieved secret words; the optimal position path, whose maximum absolute position offset is minimum, is found by a search algorithm from the constructed word index table. In this paper, a minimizing maximum weight (MMW) algorithm is employed to find an optimal position path.

Step 3: Secret message compressing. The optimal position path is converted into a binary position offset sequence. Since not all secret words appear in a text at the same time, the unretrieved secret words, which are not included in the retrieved text, are also converted into a binary sequence by a general compression algorithm, e.g., Huffman coding. The two binary sequences are linked to a complete compressed message.

Step 4: Cover text selection. Sorting the retrieved texts by the length of the compressed message in descending order, the top 10 texts are selected as the candidate cover texts.

(2) Stego text generation

With the input of 10 selected cover texts and their corresponding compressed secret message, the stego text generation obtains 10 candidte stego texts and selects a best one for high security. As a result of stego text generation, a stego text with low statistical distortion is obtained, enhancing the security of the embedded secret message.

Step 1: For the 10 candidate cover texts, the corresponding compressed secret messages are embedded into them by the synonym substitutions to generate 10 candidate stego texts respectively.

Step 2: Among the 10 candidate stego texts, the stego text with the smallest distance to its cover text is selected as the final stego text.

(3) Secret message extraction

The final stego text is transmitted to the receiver. The process of the secret message extraction will extract the secret message correctly by the following steps.

Step 1: The synonyms in the stego text are recognized by searching the synonym dictionary, which is the same as the one used by the sender.

Step 2: A binary sequence is obtained by decoding the recognized synonyms. Afterwards, the binary sequence is decompressed by the inverse process of the secret message compression. According to the obtained compressed message and the corresponding text, part of secret words can be retrieved by the decompressed positions information, and the rest part is decompressed by the compression algorithm used in the sender part.

3 Secret message compression based on word indexing

A secret message compression method is proposed to reversibly and losslessly compress the secret message into a shorter sequence, such that the subsequent embedding operations can be reduced. Under the same conditions, the shorter secret message can bring fewer changes into the cover object.

3.1 Word indexing

The secret message and the cover texts will be broken up into tokens, and the easy way is to segment them into words. Our method(WIC) does not require token normalization, stemming, and lemmatization. The tokens in the secret message should match those in the retrieved texts. All letters in a token are case insensitive, and lower case for the general case. Special characters such as punctuations are treated as separate tokens. We name the tokens of the secret message as secret words.

An exhaustive search is employed to find all positions of each secret word within a cover text. The index table is built for each text and the secret message, where each secret word points to a list of their positions in a cover text (position list).

If the text database is huge enough, there should be at least one text that contains all secret words. However, the text database in the real applications cannot cover an unlimited amount of texts. Thus, only parts of secret words will appear in one text.

Table 1 shows an example of the index table. Suppose the secret message is “changsha cloud computing and security magazine”. “changsha” and “magazine” do not appear

Table 1 An example of the index table

Secret words	Position list							
changsha	/							
cloud	3555	12480	27480	.....	49249	.....		
computing	7620	.....	14518	21199	21236	.....	.....	.....
and	22	78	.....	7609	7852	.....	16258	.....
security	1895	.....	7650	.....	16790	.....	36000	.....
magazine	/							

in the retrieved text. They are unretrieved secret words, while “cloud”, “computing”, and “security” are retrieved ones.

With the position information, the retrieved secret words can be retrieved from the cover text. We define the position offset as the word distance of two adjacent retrieved secret words in the cover text. As for the English language, it makes the average word length be 5.1 letters [3], i.e. about 40.8 bits. The maximum of a 16 bits position offset (including a sign bit) can be $2^{15} - 1$, which is very large. Predictably, the average bit length of a position offset should be greatly lower than 40. A position offset requires less space than a long secret word. If the retrieved secret words may be clustered in the cover text, the position offsets will be very small. Thus the retrieved secret words can be effectively compressed by transforming them into position offsets.

A retrieved word may appear in a text several times, while only one position of each retrieved word is selected to form a complete position path and to retrieve the secret words. Thus there exist multiple paths for the same retrieved secret words and the same cover text. In order to efficiently compress the secret message and make few modifications to the cover text, the next section gives an optimal position path searching algorithm. Note that any punctuation is regarded as an unretrieved secret word.

3.2 Optimal position path searching

For the retrieved secret words in an index table, a layered directed graph is constructed to search the optimal position path.

1) The positions in the index table are regarded as vertices.
2) The positions of one secret word are drawn in the same horizontal layer. The positions of the first secret word are arranged in the first layer; those of the second secret word are in the second layer, and so on. For two adjacent layers, each vertex of one layer is connected to all the vertices of the other layer by edges, as shown in Fig. 2, where w_i is the i th secret word, m is the total number of the secret words, p_{ij} records the j th position of w_i appearing in the retrieved text, n_i denotes the count of w_i in the retrieved text.
3) The weight assigned to each edge is the absolute position offset between the corresponding two vertices.
4) Two virtual vertices are created: the source vertex s and the end vertex t. s is connected to the vertices in the first layer by virtual edges, while t is connected to the last layer. The weight of a virtual edge is supposed to be zero.

From the source s to the end t, there are a large number of paths; the path visiting only one vertex of each layer is called the position path, which can be employed to retrieve

the retrieved secret words. Since different position paths will lead to different compression ratios, and then make different embedding efficiencies to the stego texts, an optimal problem should be considered to find the best path. The embedding efficiency [10] is a quantitative index to measure the embedding distortion, which is defined as the average number of bits embedded per one embedding change. In this paper, the compression ratio is defined to the ratio of the bit length of the compressed secret message to that of the original secret message.

In order to find the optimal position path, it is natural to consider the bit length of the maximum edge weight of each path, which determines the compression ratio. So, making use of the binary search and the Breadth-First-Search, we employ a minimizing maximum weight (MMW) algorithm to find the minimum of the maximum weight of a position path in the layered directed graph. The details of MMW algorithm is described as follows.

Algorithm 1 Minimizing maximum weight algorithm

Input: A weighted graph $G(V, E)$;

Output: The minimum of the maximum weight of a position path.

1: Calculate the maximum weight of all the edges in the graph, and denote it as R. Denote the minimum weight as L, and $L = 0$.
2: Calculate $g = \lfloor \frac{L+R}{2} \rfloor$.
3: Create an empty queue Q.
4: Enqueue (add) the vertices in the first layer into Q.
5: While Q is not empty, dequeue a vertex from Q as the current vertex cv. For each neighbor vertex nv of cv in the next layer, denote the weight of the edge between nv and cv as $weight(cv, nv)$; if $weight(cv, nv) \leq g$, then enqueue nv into Q.
6: If cv is the end vertex t, then there exists at least one path whose maximum weight is less than or equal to g, so we set $R = g$; else set $L = g$.
7: Since $L < g \leq R$, if $R - L = 1$, $g = R$, which is the minimum of the maximum weight in each path from the source to the end, and exit; otherwise return to Step 2.

Using MMW, there may be more than one path with the minimum maximum weight. In this case, we randomly choose one as the target path.

For the example in Table 1, WWM algorithm calculates the minimum maximum weight to be 2038, and finds an optimal position path "cloud: 12480, computing: 14518, and: 16258, security: 16790".

3.3 Secret message compressing

In order to effectively compress the retrieved words, the position offsets (edge weights) instead of positions in the found optimal position path are converted into a binary string. The unretrieved words can be compressed by general text compression methods such as Huffman or LZW codings. Since LZW coding takes advantage of the recurring patterns in the message to compress and save spaces, it does not perform well on the short secret message. Huffman coding is better than LZW coding for compressing the unretrieved secret words.

Compressing the whole secret message requires some additional information to distinguish the retrieved secret words from the unretrieved ones. As a result, the secret message is compressed into a binary string including following four fields.

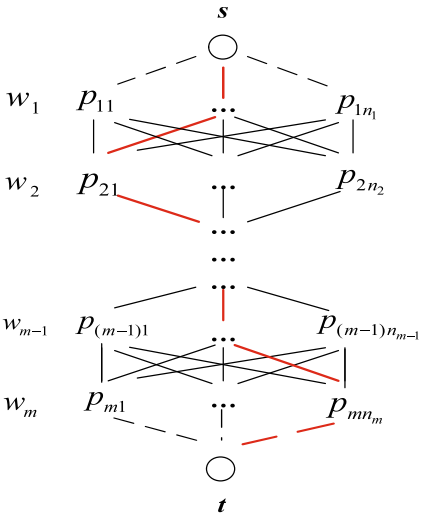


Fig. 2 The layered directed graph constructed for an index table

1). Number of the secret words (16 bits in default) recording how many words that the secret message includes. Denote the value of this field as decimal number N, such that the maximum number of the secret words can achieve to $2^N - 1$. For the example in Table 1, the value of this field is "0000 0000 0000 0110", and $N = 6$.

2). Flags (N bits) identifying whether a secret word is retrieved or not. Its length equals to the number of the secret words. If the ith secret word is a retrieved one, then the ith bit of this field is set to '1'; otherwise, it is '0'.

There are two special cases to update the value of flags.

Case 1:

If a retrieved secret word is a synonym, it is theoretically possible that its positions in the cover text may be changed, when the retrieved secret is substituted by its synonymous word in the embedding process. In this case, the secret word decompressed from the stego text will be slightly different from the original cover text. To avoid this situation, a retrieved secret synonym is treated as an unretrieved secret word, thus, its flag is set to '0'.

Case 2:

Suppose that a retrieved secret word is compressed into L_r bits, and L_u bits when it is treated as an unretrieved secret word, if $L_r > L_u$, then treating it as an unretrieved one is beneficial for improving compression performance. In this case, the corresponding flag bit will be set to '0'.

The example in Table 1 has four retrieved secret words and two unretrieved ones, thus the flag field is initialized to "011110". By checking the synonym dictionary, we find that there are no synonyms in the secret message, so no updates are required by considering the special case 1. The judgment of special case 2 requires the compression result of the retrieved secret words and the unretrieved secret words. Thus, the flags should be updated at last.

3). Compressed retrieved secret words MMW algorithm finds a position path for losslessly compressing the retrieved secret words. The first position information in the path may be far greater than the maximum of the absolute position offsets. So, it will be solely encoded instead of being regarded as a start position offset. This field includes four sub fields.

(a) The bit length of the first position information(8 bits in default): 8 bits is employed to allocate enough space to encode the first position. Denote the value of this field as the decimal number L_f.
(b) The first position (L_f bits): encoding the first position information into L_f bits string.
(c) The maximum bit length of position offsets (5 bits in default): identifying the maximum bit length of the position offsets in the optimal position path. Denote the value of this field as the decimal number L_r.
(d) The position offset list ($(N_r - 1) \times L_r$ bits): Each position offset is allocated a size of L_r bits. The most significant bit is the sign bit; '0' represents a positive position offset, while '1' represents a negative one. N_r retrieved secret words require $(N_r - 1) \times L_r$ bits to encode all position offsets.

For the example in Table 1, by using the MMW algorithm, the position path for the retrieved secret words is "cloud: 12480, computing: 14518, and: 16258, security: 16790". Then the offsets are "2038, 1740, 532"; the maximum absolute offset is 2038; the maximum bit length of the offsets(including a sign bit) is 12; the bit length of the first position is 14. Thus, the retrieved secret words are converted into the following binary string, whose total length is 63.

$$\underbrace{value}_{length}\ \underbrace{00001110}_{8}\ \underbrace{11000011000000}_{14}\ \underbrace{01100}_{5}\ \underbrace{0\ 11111110110}_{12}$$

$$\underbrace{0\ 11011001100}_{12}\ \underbrace{0\ 01000010100}_{12}$$

4). Compressed unretrieved secret words All unretrieved secret words are concatenated into a character string with the spaces as the delimiters. N_u unretrieved secret words requires N_u spaces, while the last space is employed to represent the end. Each character is encoded into variable length bits by Huffman coding.

In the example, the string converted from the unretrieved secret words in lowercase is "changsha magazine", with an extra space at the end. Suppose that the character set just contain 26 lowercase letters, space, and 11 most frequently used punctuations, by counting their frequencies in a corpus, we can construct a Huffman tree and get the codeword of each character. As a result, "changsha magazine" is encoded by Huffman Coding into the following binary string, whose bit length is 83.

$$\underbrace{value}_{character}\ \underbrace{00010}_{c}\ \underbrace{0100}_{h}\ \underbrace{1010}_{a}\ \underbrace{1000}_{n}\ \underbrace{011011}_{g}\ \underbrace{0101}_{s}\ \underbrace{0100}_{h}\ \underbrace{1010}_{a}$$

$$\underbrace{111}_{space}\ \underbrace{110111}_{m}\ \underbrace{1010}_{a}\ \underbrace{011011}_{g}\ \underbrace{1010}_{a}\ \underbrace{01101000101}_{z}\ \underbrace{0111}_{i}\ \underbrace{1000}_{n}$$

$$\underbrace{001}_{e}\ \underbrace{111}_{space}$$

At last, the Flags should be updated by considering the special case 2. Each retrieved secret word together with a space is encoded by Huffman coding. If the length of its codeword is smaller than the maximum bit length of the position offsets, the corresponding Flag bit '1' is updated to '0'. In the example, the four retrieved secret words all have longer codewords when they are encoded by Huffman coding. Therefore, the Flags do not need be updated.

In conclusion, by the proposed compression method (WIC), the 368 bits secret message in the example is compressed into a 168 bits binary string, thus the compression ratio is 45.65%. For comparison, the whole secret message is directly encoded by Huffman coding and LZW coding, respectively. By Huffman coding, the compression ratio is 57.34%. By LZW coding, the compression ratio is 67.39%. It is easy to find that our compression method can greatly reduce the length of the practical embedding information.

4 Stego text generation and secret message extraction

The security of the steganography has relations with the length of the embedding information and the modifications on the statistical features of the cover text. The proposed compression method can achieve different compression ratios with different cover texts even for the same secret message. For the given text database, ten texts are selected as candidate cover texts by comparing their compression ratios, consequently, ten candidate stego texts are generated. A selection method is designed to select the best stego text in this Section.

4.1 Synonym substitution based message embedding

A synonym dictionary consisting of synonym sets is employed for steganography. The synonymous words are arrayed in a synonym set and encoded as a unique binary digit.

For a given synonym set $S = \{s_0, s_1, \ldots, s_{n-1}\}$, let the n synonymous words be in descending order in terms of their frequencies. The synonym s_i is encoded by the rule shown in (1).

$$e(s_i) = \begin{cases} 0 & i = 0 \\ 1 & else \end{cases} \tag{1}$$

where $e(s_i)$ represents the encoded value of s_i. The word with the highest relative frequency is encoded as '0', viz. $e(s_0) = 0$, while all the others are denoted as '1'.

Synonym substitution rule Suppose that the embedded bit is b, the corresponding cover synonym is s, which is located in the synonym set $S = \{s_0, s_1, \ldots, s_{n-1}\}$; if $e(s) = b$, no substitution is done; if $e(s) \neq b$, a synonym $s^{'}$ is randomly selected from S to replace s and make $e(s^{'}) = b$.

In a stego text, if there is a synonym s, then the embedded bit $b = e(s)$ is extracted.

4.2 Stego text selection

By employing the above synonym substitution rule, ten candidate stego texts can be generated for each secret message. The stego text, whose statistical features are closest to those of its corresponding cover text, is finally selected. Thus, it is difficult to distinguish the final stego text from the cover texts.

In order to measure the statistical distance between a stego text and its cover text, both of them are represented into the statistical features similar to those in a steganalysis method. Suppose that the feature set of the stego text is $V^s = \{v_0^s, v_1^s, \ldots, v_{n-1}^s\}$, and the one of the cover text is $V^c = \{v_0^c, v_1^c, \ldots, v_{n-1}^c\}$. The Euclidean distance is employed to measure the

distance between the stego text and its cover text, i.e., the distance between vectors V^s and V^c is defined as follows.

$$d(V^s, V^c) = \sqrt{\sum_{i=0}^{n-1}(v_i^s - v_i^c)^2} \tag{2}$$

Denote the feature vector of the jth candidate stego text as $V_j^s = \{v_{j0}^s, v_{j1}^s, \ldots, v_{j(n-1)}^s\}$, and its corresponding cover text as $V_j^c = \{v_{j0}^c, v_{j1}^c, \ldots, v_{j(n-1)}^c\}$, then their distance is $d(V_j^s, V_j^c) = \sqrt{\sum_{i=0}^{n-1}(v_{ji}^s - v_{ji}^c)^2}$. Then, the final stego text $S_{\widehat{j}}$ with the shortest distance can be selected by (3).

$$S_{\widehat{j}} = \arg\min_j d(V_j^s, V_j^c) \tag{3}$$

4.3 Secret message extraction

The final stego text will be sent to the receiver, and the secret message can be extracted from the received stego text. (1) is used to calculate the encoded values of synonyms in the stego text; then the obtained binary string is decompressed by an inverse process of the compression to recover the original secret message. The details of decompressing the embedded message can be described as follows.

Algorithm 2 Secret message decompression algorithm

Require: binary string $M^{'}$ obtained by decoding the synonyms in the stego text $C^{'}$, Huffman tree used by the sender.

Ensure: Secret message M.

1: Convert the first 16 bits of $M^{'}$ to a decimal number N.
2: Read the next N bits of $M^{'}$, and denote it as $F = F_1 F_2 \cdots F_N$. Calculate the number of bit '1' and '0' in F, and denote them as N_r and N_u, respectively.
3: Read and convert the next 8 bits of $M^{'}$ to a decimal value L_f.
4: Read and convert the next L_f bits to a decimal value p_1. Index stego text $C^{'}$ to get the p_1th words rw_1.
5: Read and convert the next 5 bits to a decimal value L_r.
6: Set i=2.
7: Read and convert the next L_r bits to a decimal value o_i with a sign bit; calculate $p_i = p_{i-1} + o_i$. Finally, index the stego text $C^{'}$ to get the p_ith words rw_i.
8: If $i < N_r$, $i = i + 1$, and goto Step 7; else goto next step.
9: Read the remaining bits of $M^{'}$ and traverse the Huffman tree node by node as each bit to obtain the decompressed characters. By taking the space as the delimiter, the first N_u decompressed words $\{uw_j\}$ are extracted.
10: Set $i = 1$, $j = 1$, $k = 1$, $M =^{'}\ ^{'}$.
11: If $F_i = 0$, then select the jth words rw_j appending to M, $M = M + rw_j + \text{' '}$, $j = j+1$, $i = i + 1$; else select the kth words uw_k appending to M, i.e., $M = M + uw_j + \text{' '}$, $k = k + 1$, $i = i + 1$.
12: Repeat Step 10 until $i = N$.
13: **return** M.

As an example, let us consider the compressed secret message in Section 3.3.

1) After decoding the values of synonyms in the stego text, we get a binary string 0000000000000110 011110 0000 1110 11000011000000 01100 011111110110 011011001100 001000010100 00010 0100 1010 1000 011011 0101 0100 1010 111 110111 1010 011011 1010 01101000101 0111 1000 001 111
2) From the first 16 bits, we have a decimal value 6.
3) From the next 6 bits, we have $F = 011110$, $N_r = 4$, and $N_u = 2$.
4) From the next 8 bits, we have a decimal value 14.
5) From the next 14 bits, we have $p_1 = 12480$. By indexing the stego text, we find the 12480^{th} words $rw_1 = cloud$.
6) From the next 5 bits, we have a decimal value 12.
7) From the next $12 * 3$ bits, we get 2038, 1740, 532, for every 12 bits. Thus, we have $p_2 = 14518$, $p_3 = 16258$, $p_4 = 16790$. By indexing the stego text, we find the 14518^{th}, 16258^{th}, 16790^{th} words, thus, we have $rw_2 = computing$, $rw_3 = and$, $rw_4 = security$.
8) Read the remaining bits and traverse the Huffman tree node by node until the second space is obtained. The decoded characters are "changsha magazine", i.e., $uw_1 = changsha$, $uw_2 = magazine$.
9) Set $i = 1$, $j = 1$, $k = 1$.
10) As $F_1 = 0$, $M = M + uw_1 + ' ' = changsha$; As $F_2 = 1$, $M = M + rw_1 + ' ' = changsha\ cloud$; As $F_3 = 1$, $M = M + rw_2 + ' ' = changsha\ cloud\ computing$ As $F_4 = 1$, $M = M + rw_3 + ' ' = changsha\ cloud\ computing\ and$; As $F_5 = 1$, $M = M + rw_4 + ' ' = changsha\ cloud\ computing\ and\ security$; As $F_6 = 0$, $M = M + uw_2 + ' ' = changsha\ cloud\ computing\ and\ security\ magazine$. The final recovered secret message is "changsha cloud computing and security magazine".

5 Experimental results and analysis

In this section, we give the experimental results, which demonstrate the efficiency of the proposed steganographic method, especially in terms of the compression method and the stego text selection strategy.

5.1 Experimental setup

Compared to common steganography, the proposed steganography requires a text database instead of a single cover text, special secret messages consisting of natural language words, and a preprocessed synonym database for encoding and decoding synonyms. In our experiments, 10000 English free ebooks, which were downloaded from the Project Gutenberg (http://www.gutenberg.org/), compose of the text database. They are all famous literatures. Three secret message sets Set1, Set2, Set3 were built, each of which consists of 100 messages copied from the News on the Internet. The average lengths of the secret messages in these three sets are about 100, 200, 500 bytes, respectively. In order to verify the impact on compression performance caused by the source of secret message, we selected 200 secret messages from other ebooks of Gutenberg corpus. The 200 messages were divided into Set4 and Set5. The average lengths of the messages in Set 4 and Set5 are about 100 and 200 bytes respectively. Finally, the proposed method utilizes the synonym database provided by [19].

In order to improve the run speed, we preprocessed the text database before searching an optimal position path by MMW. For each secret message, we counted the number of the retrieved secret words in each text, sorted the retrieved texts in descending order, and then filter the last some texts out. The rest of the retrieved texts contains much more retrieved secret words, resulting in much lower compression ratios. The proposed compression method WIC only performed on the selected texts.

For comparison, Huffman coding and LZW coding were employed to compress the secret message. Meanwhile, a compression method called Random Algorithm (RA) was designed, which is similar to the proposed WIC method. The only difference is that RA randomly selects a position path to represent the retrieved secret words.

5.2 Compression ratio

This experiment compares the compression ratios of four compression methods including RA, Huffman, LZW coding, and WIC. The compression ratios of RA and WIC were calculated over the 5000 candidate cover texts with 500 secret messages in Set1 to Set5. Each secret message has ten candidate compression ratios, but only the minimum was selected for comparison. Huffman and LZW codings directly compressed the 500 secret messages to calculate their compression ratios. The minimum, the maximum, and the average of the compression ratios of each method for different secret message sets are listed in Table 2. And the detailed results of the arbitrary selected 100 secret messages are illustrated in Fig. 3.

Table 2 and Fig. 3 show that the compression ratios of the proposed WIC are lower than those of three comparison methods in all cases. Our method can make less bits be embedded into the cover text, that is the embedding capacity is increased. As WIC and RA both benefit from the cover text to compress the secret message, RA has the proximate compression ratios. From Table 2 and Fig. 3, it can also be found that RA performs better than Huffman and LZW codings, while Huffman coding compresses the secret message shorter than LZW coding does. Besides, all the methods provide stable compression performance as the length of the secret message increases.

Table 2 Compression ratios comparison

		Set1	Set2	Set3	Set4	Set5
Huffman	minimum	0.4952	0.5068	0.5144	0.4764	0.4646
	maximum	0.5719	0.5683	0.5419	0.5680	0.5757
	average	0.5268	0.5281	0.5270	0.5201	0.5270
LZW	minimum	0.6250	0.6131	0.6210	0.6332	0.6238
	maximum	0.7936	0.7960	0.7331	0.7962	0.7843
	average	0.7176	0.7236	0.6886	0.7180	0.7289
RA	minimum	0.3534	0.3564	0.3575	0.3377	0.3541
	maximum	0.5235	0.5093	0.4851	0.5140	0.5014
	average	0.4513	0.4400	0.4392	0.4368	0.4307
MMW	minimum	0.3125	0.3198	0.3235	0.1310	0.1231
	maximum	0.4988	0.4517	0.4563	0.4352	0.4321
	average	0.3902	0.3885	0.3964	0.3784	0.3837

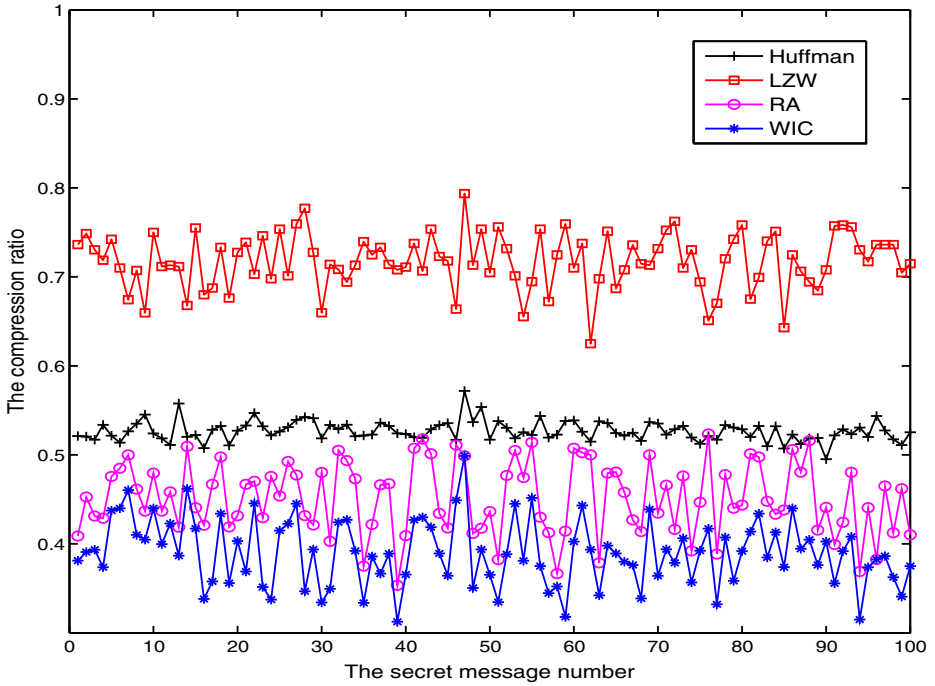


Fig. 3 Parts of the compression ratios for comparison

It is worth noting that the average compression ratios of WIC in Set4 and Set5 are slightly lower that those in Set1, Set2 and Set3. Particularly, the minimums in Set4 and Set5 are much less than those in Set1, Set2 and Set3. Although the secret messages in Set4 and Set5 are not captured from the prepared retrieved texts, they are all from the same Gutenberg Corpus. There is very likelihood that the words of a secret message in Set4 or Set5 concentrates in some of the retrieved texts in the text database. Extremely, a retrieved text may the same content with the complete secret message. As consequence, the minimum maximum weight of the found optimal position path will be reduced, evenly it is very small, thus the average compression ratio will be reduced. In some extreme cases, the compression ratio will greatly fall to a very low value. The smaller the compression ratio is, the larger the relative embedding capacity achieves. Namely, for the same cover text, more space will be saved for embedding longer original secret message if the secret message can be efficiently compressed into a shorter message.

5.3 Embedding efficiency

The compression of the secret message can improve the embedding efficiency of the steganography by reducing the number of modifications, as the practical embedded payload is decreased. By using the synonym substitution rule (see Section 4.1), 500 compressed secret messages of each compression method were separately embedded into 5000 candidate cover texts, which were selected by WIC. And the original 500 secret messages were also directly embedded to generate 5000 candidate stego texts denoted as NOC stego texts.

For the candidate stego texts from different secret message sets, the minimum, the maximum, the average of embedding efficiencies of each compression method are respectively calculated, and are listed in Table 2. And the detailed results of the arbitrary selected 100 stego texts are illustrated in Fig. 4.

Suppose that the secret message is a random sequence, then the encoded value of a synonym has a probability of 0.5 to equal the embedded bit, as one bit is embedded into one cover synonym. The embedding efficiency for NOC stego texts is nearly 2 (bit per synonym substitution), which is demonstrated by the results listed in Table 3. However, more than one secret message bit will be embedded into one cover synonym after being compressed, as the embedded compressed message is shorter than the original secret message. Consequently, the average embedding efficiency of the compression-based steganography is larger than 2 demonstrated by the results in Table 3. Particularly, that of WIC-based method can achieve 4.5. In extreme cases, the maximum of its embedding efficiency can achieve 15.6981, which is far larger than those of other methods, as the corresponding secret message has be compressed effectively with a very low compression ratio. The experimental results show that the proposed WIC-based method has the higher embedding efficiency than other compared methods. Namely, when the same cover text is embedded the same secret information, the change of statistical characteristics made on the cover text by the proposed method would be minimal.

From Figs. 3 and 4, it can be found that the compression ratio greatly affects the embedding efficiency. A low compression ratio may result in a high embedding efficiency, thus our WIC-based method with better compression performance has the higher embedding

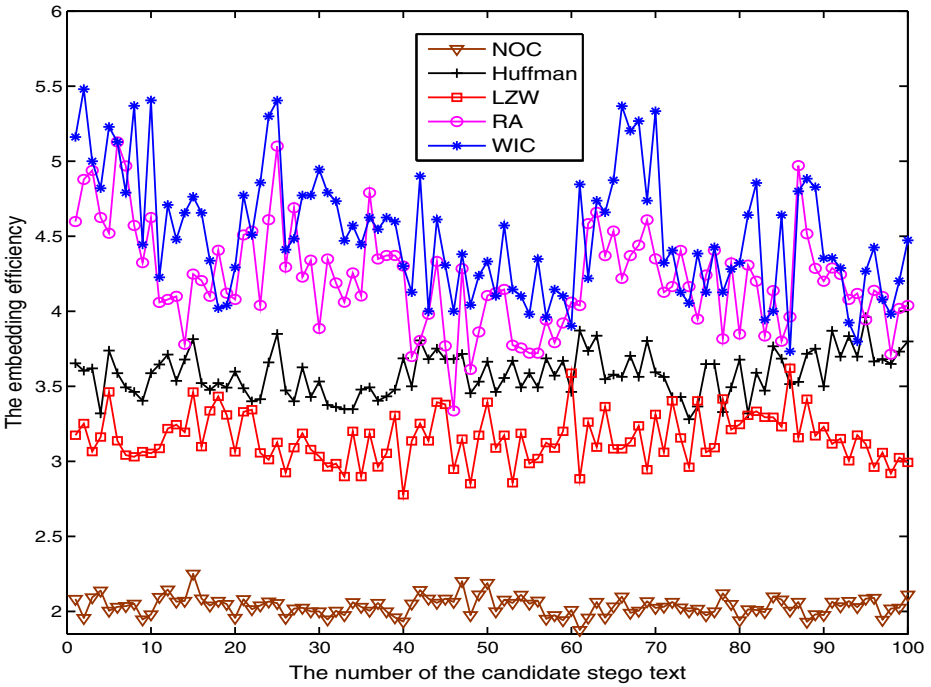


Fig. 4 Parts of the embedding efficiencies for comparison

Table 3 Embedding efficiency comparison

		Set1	Set2	Set3	Set4	Set5
NOC	minimum	1.7058	1.8581	1.8613	1.7325	1.5767
	maximum	2.2921	2.2005	2.1779	2.3415	2.9275
	average	2.0335	2.0362	2.0347	2.0533	2.0541
Huffman	minimum	2.9904	2.9963	3.1574	2.4425	3.037
	maximum	4.2041	3.8935	3.8767	4.8282	4.3011
	average	3.5470	3.5205	3.5202	3.5694	3.5464
LZW	minimum	2.1390	2.7248	2.8965	2.3774	2.2112
	maximum	4.1224	4.0100	4.0200	4.2812	3.9649
	average	3.1950	3.2715	3.4065	3.2098	3.2458
RA	minimum	3.2308	3.4262	3.5840	2.795	3.4639
	maximum	5.9429	5.6741	6.2112	6.222	5.7183
	average	4.1860	4.2888	4.3050	4.3142	4.3756
WIC	minimum	3.4335	3.5850	3.8902	3.6151	3.4681
	maximum	6.2769	6.2756	6.0698	15.698	14.579
	average	4.5464	4.5856	4.5652	4.7813	4.6618

efficiency than RA-base method, while Huffman coding based method performs better than LZW-based method in terms of embedding efficiency and compression ratio.

The matrix encoding based steganographic method in [21] can embed k bits into $2^k - 1$ cover synonyms by substituting one synonym at most, thus its embedding efficiency e in theory is $(k \times 2^k)/(2^k - 1)$. $k = 4$, $e \approx 4.267$, while $k = 5$, $e \approx 5.161$. The compression-based steganography can further improve the embedding efficiency by using matrix encoding to embed the compressed message. Therefore, cooperating WIC with matrix encoding, the average embedding efficiency can be significantly exceed 4.5 in the future work.

5.4 The capability of anti-steganalysis

In this section, the anti-steganalysis capabilities of the linguistic steganographic methods is tested using the linguistic steganalysis method presented in [6], as they are employed to measure the security of the secret message. Here, the detection precision defined as the ratio of correctly predicted stego texts to all stego texts is employed to measure the anti-steganalysis capability of a stegnographic method. The higher the detection precision is, the worse the anti-steganalysis capability of the steganographic method is. On the contrary, the lower the detection precision is, the better the anti-steganalysis capability is.

All candidate stego texts mentioned in Section 5.3 were detected. In terms of the derivation of secret messages, each category of candidate stego texts was divided into two groups. The stego texts of one group were generated by embedding secret messages in Set1, Set2 and Set3. The remaining stego texts consist of the other group. The steganalysis results listed in Table 4 demonstrate that the compression ratio and the embedding efficiency both have contributions to decrease the anti-steganalysis capability of the stego texts.

With the secret message compression technique, the compression based steganographic methods perform well in resisting steganalysis, while the NOC-based stego text has the worst anti-steganalysis capability. As the candidate cover texts are chose by the compression results of WIC, their sizes are generally large. When the messages with fixed length

Table 4 Steganalysis results for all candidate stego texts

Steganography	Set1+Set2+Set3			Set4+Set5		
	Stego text	Cover text	Precision	Stego text	Cover text	Precision
NOC	2306	694	76.87%	1535	465	76.75%
Huffman	1614	1386	53.80%	1091	909	54.55%
LZW	1429	1571	47.63%	943	1057	47.15%
RA	1329	1671	44.30%	881	1119	44.05%
WIC	1264	1736	42.13%	826	1174	41.30%

are embedded, just parts of cover synonyms in the candidate cover texts are used to be embedded message with low embedding rate. The detection capability of a steganalysis is deteriorated as the embedding rate decreases. When detected stego text is hidden with the low embedding rate, the detection precision is particularly not so good. Therefore, the precision of detecting NOC-based stego texts is lower than the related results in [6]. On the other hand, applying the compression methods not limited to WIC to other steganographic methods, their capabilities of resisting steganalysis will be also improved by compressing the secret message before embedding.

Obviously, the lower the compression ratio, the lower practical embedding payload is, the higher the embedding efficiency is, and the more difficult the detection task is. From the results in Table 4, the WIC-based stego text has best capability of resisting steganalysis in [6]. The anti-steganalysis capability of WIC-based stego texts derived from Set1+Set2+Set3 is slightly higher than that from Set4+Set5. We can see that when the retrieved texts for compression and secret messages come from the same corpus, the anti-steganalysis capability will be slightly improved. Similar observations can be made for the RA-based stego texts, which have second best anti-steganalysis capability. Whereas conversely, the LZW-based stego text performs better than Huffman -based stego text, although the former has a lower embedding efficiency than the latter. We think of that one reason can explain these observations: The compressed secret message by LZW coding is no longer independent and identically distributed, and which is similar to the encoded values of synonyms in the cover text. As a result, few synonym substitutions would be done leading to few changes in the statistical characteristics of the cover text, thus the anti-steganalysis capability of the LZW-based stego text is greatly improved. To sum-up, the compression ratio of the secret message is not the only factor that determines the capability of anti-steganalysis.

The stego text selection in Section 4.2 can efficiently improve the anti-steanalysis capability of the stego text. In this experiment, the features used in [6] and [20] were extracted

Table 5 Steganalysis results for the selected stego texts

	Stego text	Cover text	Precision
NOC	254	246	50.80%
Huffman	169	331	33.80%
LZW	135	365	27.00%
RA	138	362	27.60%
WIC	129	371	25.80%

from the text to compose a statistical feature vector, which is used for measuring the distance (by Eq. (2)) between a stego text and its corresponding cover text. By the stego text selection, 500 stego texts were selected from 5000 candidate stego texts for each category. The detection results of the selected stego texts by the steganalyis in [6] are listed in Table 5. It can be found that the capability of resisting the steanalysis is significantly improved by the selection strategy in all cases compared with Table 4. The detection results also demonstrate once again that the WIC-based stego text has the higher security compared with the stego texts of other four categories.

6 Conclusions

In this paper, we have presented a new linguistic steganography, which includes two important innovations: secret message compression and stego text selection. Different from existing compression methods, the proposed secret message compression method MIC considers the cover text to decrease the compression ratio, and then produces 10 candidate stego texts. The practical embedded payload is greatly reduced by compressing the secret message, thus, the security of the secret message in a stego text is improved by reducing embedding modifications measured by the embedding efficiency. The final stego text is selected by measuring the distance between the features of each candidate stego text and its cover text, further improving the undetectability of the secret message. The experimental results demonstrate that the proposed method outperforms existing state-of-the-art methods, and it can hide more secret messages and secure them well.

The proposed WIC has quite different requirements compared to the normal compression works. It is special effective for compressing the natural language words. However, WIC cannot work well without a comprehensive text database. In general, the larger the text database, the better the compression performance will achieve. However, it is still questionable whether this method can be applied for image steganography, and it could be further research.

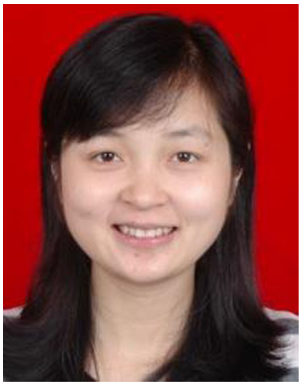
In future work, our intention is also to minimize both the linguistic and statistical distortion caused by secret message embedding. We can incorporate word embeddings generated by Neural Network Language Model for measuring linguistic distortion, and combine the WIC method with Syndrome-trellis Code to minimize the statistical distortion. Finally, in order to have a better understanding of the difference between a cover and stego text, theoretical investigations will be considered to present new stego text selection methods with minimized distortion.

Acknowledgements This work has been performed in the Project "Research on Theory and Approach of Secure Text Steganography" supported by National Natural Science Foundation of China (No. 61202439), and partly supported by National Natural Science Foundation of China (No. 61302159).

Compliance with Ethical Standards The authors declare that they have no conflict of interest. All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards. This article does not contain any studies with animals performed by any of the authors. Informed consent was obtained from all individual participants included in the study.

References

1. Atallah MJ, Raskin V, Hempelmann C, Karahan M, Sion R, Topkara U, Triezenberg KE (2002) Natural Language Watermarking and Tamperproofing. In: The 5th International Workshop on Information Hiding. Springer-Verlag, Berlin, pp 196-212
2. Bergmair R (2007) A comprehensive bibliography of linguistic steganography. In: Proceeding of SPIE 6505, Security, Steganography, and Watermarking of Multimedia Contents IX, pp 65050W
3. Bochkarev VV, Shevlyakova AV, Solovyev VD (2012) Average word length dynamics as indicator of cultural changes in society. Computer Science. arXiv:1208.6109
4. Bolshakov IA (2004) A method of linguistic steganography based on collocationally-verified synonymy. In: Information Hiding. Springer, Berlin, pp 180–191
5. Chang CY, Clark S (2014) Practical linguistic steganography using contextual synonym substitution and a novel vertex coding method. Comput Linguist 40(2):403–448
6. Chen Z, Huang L, Yang W (2011) Detection of substitution-based linguistic steganography by relative frequency analysis. Digit Investig 8(1):68–77
7. Crandall R (1998) Some notes on steganography. http://dde.binghamton.edu/download/Crandall_matrix.pdf. Accessed 1 August 2017
8. Denemark T, Fridrich J (2017) Steganography with multiple JPEG images of the same scene[J]. IEEE Trans Inf Forensic Secur 12(10):2308–2319
9. Fridrich J, Soukal D (2006) Matrix embedding for large payloads. IEEE Trans Inf Forensic Secur 1(3):390–395
10. Fridrich J, Goljan M, Soukal D (2006) Wet paper codes with improved embedding efficiency. IEEE Trans Inf Forensic Secur 1(1):102–110
11. Hu X, Luo G, Lu Y, Xiang L (2013) A steganography on synonym frequency distribution. Adv Inf Sci Serv Sci 5(10):206–214
12. Li J, Huang X, Li J, Chen X, Xiang Y (2014) Securely Outsourcing Attribute-based Encryption with Checkability. IEEE Trans Parallel Distrib Syst 25(8):2201–2210
13. Li J, Yu C, Gupta BB, Ren X (2018) Color image watermarking scheme based on quaternion Hadamard transform and Schur decomposition. Multimed Tools Appl 77(4):4545–4561
14. Liu YL, Sun X, Gan C, Wang H (2007) An efficient linguistic steganography for Chinese text. IEEE International Conference on Multimedia and Expo. IEEE, pp 2094–2097
15. Meral HM, Sankur B, ozsoy AS, Gungor T, Sevinc E (2009) Natural language watermarking via morphosyntactic alterations. Comput Speech Lang 23(1):107–125
16. Muhammad HZ, Rahman SMSAA, Shakil A (2009) Synonym based Malay linguistic text steganography. Innovative Technologies in Intelligent Systems and Industrial Applications, CITISIA 2009. IEEE, pp 423–427
17. Topkara U, Topkara M, Atallah MJ (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. ACM, pp 164–174
18. Westfeld A (2001) F5-a steganographic algorithm high capacity despite better steganalysis. Lect Notes Comput Sci 2137:289–302
19. Winstein K (1998) Lexical steganography through adaptive modulation of the word choice hash. http://www.imsa.edu/?keithw/tlex. Accessed 2 August 2017
20. Xiang L, Sun X, Luo G, Xia B (2014) Linguistic steganalysis using the features derived from synonym frequency. Multimed Tools Appl 71(3):1893–1911
21. Yang X, Li F, Xiang L (2015) Synonym Substitution-based Steganographic Algorithm with Matrix Coding. J Chin Comput Syst 36(6):1296–1300
22. Yu C, Li J, Li X et al (2018) Four-image encryption scheme based on quaternion Fresnel transform, chaos and computer generated hologram[J]. Multimed Tools Appl 77(4):4585–4608



Lingyun Xiang received her B.E. in computer science and technology, in 2005, and the Ph.D. in computer application, in 2011, Hunan University, Hunan, China. Currently, she is a Lecturer at School of Computer and Communication Engineering, Changsha University of Science and Technology. Her research interests include information security, steganography, steganalysis, machine learning, and pattern recognition.

Wenshuai Wu is pursuing his M.E. in communication and information system from Changsha University of Science and Technology. He received his BE in network engineering from Changsha University of Science and Technology, China, in 2014. His research interests include information security and steganography.

Xu Li received the master's degree in probability theory from the College of Mathematics and Computing Science, Changsha University of Science and Technology, Changsha, China, in 2006. He received his Ph.D. degree in Computer Application from Hunan University, Changsha, China, in 2016. Currently, He is a Lecturer at School of Mathematics and Statistics, Changsha University of Science and Technology. His research interests include probability theory, image processing, image forensics, and steganography.

Chunfang Yang received the B.S., M.S., and Ph.D. degrees from the Zhengzhou Information Science and Technology Institute in 2005, 2008, and 2012, respectively. Currently, he is a lecturer with Zhengzhou Science and Technology Institute. His current research interests include image steganography and steganalysis technique.