Journal of Computers Vol. 30 No. 3, 2019, pp. 28-43
doi:10.3966/199115992019063003003

The Implementation of a Secret Information Hiding Framework Based on Hash Function and Invisible ASCII Characters Replacement

Er-Zhou Zhu[1], Zhu-Juan Ma[2*], Hui Sun[1], Feng Liu[1]

[1] School of Computer Science and Technology, Anhui University, Hefei 230601, China
{ezzhu, sunhui, fengliu}@ahu.edu.cn

[2] School of Economic and Technical, Anhui Agricultural University, Hefei 230036, China
zjmsjtu16@gmail.com

Received 10 August 2017; Revised 10 January 2018; Accepted 1 March 2018

Abstract. As the text document is widely and frequently used in the communication, the text-based information hiding technology is still an important topic in computer security. However, the current text-based information hiding techniques are suffered from some problems, such as poor robustness, lower embedding rate and lower efficiency. Aiming at these problems, this paper proposes a novel secret information hiding framework by integrating of the hash function and the invisible ASCII character replacement technology. Under this framework, the encoded secret information is firstly divided into groups with even number of “1”. Secondly, the *SP*s (space characters) in each carrier segment are replaced with *SOH* (start of head) by our replacement algorithm. Thirdly, the replaced segment is processed by hash function. Finally, the information hiding process is completed by comparing the generated hash values with the encoded secret information. Furthermore, by uti-lizing the hash collisions of the previous segments, the framework is improved to enhance the security and the embedding rate. The experimental results have demonstrated that the proposed framework holds high em-bedding rate, strong robustness and high efficiency.

Keywords: character replacement, hash function, information security, text-based information hiding

1 Introduction

Nowadays, the safety transmission of private information by the Internet attracts much attention from both industry and research communities. In order to ensure the security of information transmission, many information hiding technologies are proposed [1-2]. In general, there are mainly two kinds of the information hiding technology, the digital watermarking [3] and the steganography [4]. Since the widely and frequently usage of the text document in the communication, many works have been deployed on embedding the secret information into the text-based documents.

According to the embedding styles, the text-based information hiding mechanisms can be divided into two types: the format-based information hiding and the content-based information hiding. Specifically, the format-based information hiding embeds the secret information into the carrier document by adjusting its font, line space, word space, words count in one line, adding blank characters and so on [5]. The content-based information hiding is realized by processing the syntax [6] and semantic [7] of natural languages. Meanwhile, most of these methods employ mathematical statistics, substitution table and machine translation strategies.

Since the underlying natural language processing technology is far from mature by now, there are still many obstacles to be resolved in this technology. On one hand, by the present language processing

[*] Corresponding Author

algorithms, there is obvious distinction between the generated documents that carrying the secret information and the original nature language documents. This difference can be easily distinguished by unaided eyes. So, the documents that generated by the present algorithms cannot meet the practical requirements from point views of syntax, semantic and statistical properties. On the other hand, the complex features of natural language make it is a difficult work for constructing effective substitution tables for the replacement-based hiding algorithms. In addition, most of the replacement-based hiding algorithms are accomplished by a simple way, i.e. “0” represents replacement and “1” represents unchanged or reverse. It is hard for this simple mode to resist attacks from detecting algorithms based on statistic analysis.

In order to cope with issues existing in the current context-based information hiding algorithms, this paper proposes an effective secret information hiding framework based on the hash function and invisible ASCII characters replacement technologies. Generally speaking, the proposed framework has the following new features:

Strong robustness. By introducing the word shift based information hiding method and setting the restriction on the number of space symbols in each carrier segment, the proposed framework brings much less “noise” to the carrier document. By the framework, the generated document exhibits nearly the same effect with the original ones. As a result, the interceptor cannot easily detect the existence of the secret information. Meanwhile, this framework needn’t to construct a substitution table, which eliminates the safety risk brought by the table.

High embedding rage. In our framework, the hash function is employed to make the generated document more difficult to be detected. However, this method brings inevitable hash collisions. Aiming at this problem, in this paper, the hash collisions of the previously processed segments are used to generate more available hash values for their subsequent segments. By doing this, the high embedding rate is achieved.

High efficiency. The traditional information hiding technology has no strict restriction on the efficiency. An information hiding algorithm is acceptable as long as it has high robustness and high embedding rate. However, with the rapid development of mobile environment, it is urgent needs “good” (with relatively lower runtime overhead) information hiding algorithms that can be deployed on the resource starvation mobile platforms. By utilizing the simple secret information embedding and extracting strategies, and the reasonable values of h (the secret information has to be split into groups with h bits) and k (there are k space symbols in a given segment of the carrier document), the runtime overhead of our framework is relatively lower than the ones of Singh [8] and Rahman [9]. The small runtime overhead makes our framework is also applicable to the mobile platforms.

This paper is a continuation of the work presented in [10]. On the basis of that paper, we have made tremendous improvements. The remainder of this paper is organized as follows. Section 2 discusses the related work on the text-based information hiding technology. Section 3 briefly analyzes the invisible ASCII characters. Section 4 discusses the implementation of the proposed framework. Section 5 gives its improvement. Section 6 evaluates the experimental results. Finally, Section 7 briefly concludes the paper and outlines our future works.

2 Related Work

As mentioned in Sect. 1, the text-based information hiding mechanisms can be divided into two types: the content-based information hiding technology and the format-based information hiding technology.

2.1 Content-based Information Hiding

The content-based information hiding technique is realized by processing the syntax and semantic of natural languages. Meanwhile, most of these methods employ the substitution table and machine translation strategies.

The syntax-oriented method utilizes the changing of the grammatical structure of nature language to generate the implicit document. As a classical work, TEXTO [11] is composed of the sentence template and the substitution dictionary. TEXTO works just like a simple substitution ciper, with each of the 64 ASCII symbols or Unicode from secret data replaced by an English word. Under this method, the secret information is replaced by English words in substitution dictionary. Wayner [12] introduces a method by per-computing the context-free grammars to generate steganographic text without sacrificing syntactic and semantic correctness. Chapman proposes another setganorgraphic method called NICETEXT [13]. Similar to TEXTO, NICETEXT needs to construct substitution table before the embedding procedure filling English words into the sentence template.

The semantic-oriented method hides the secret information by the synonym substitution. Common algorithms include replacement based on equivalent rules [14], replacement based on the synonymous sentences [15], replacement based on synonym substitution [16], and steganography based on machine translation [17].

The syntax-oriented method is easy to implement. However, the generated text by this method is not coherent to the original carrier document and does not have the semantic integrity. Therefore, it may be easily noticed by interceptors [18]. The key of semantic-oriented approach is to structure an ideal synonym substitution table by utilizing the natural language processing mechanisms. Consequently, the performances of these approaches are determined by the development of natural language processing technology [19]. Different from the above two methods, the secret information hiding framework proposed by this paper does not change the semantic and the syntax of the original cover texts. By adopting the invisible ASCII codes replacement strategy and the hash collision, the generated documents are more in accordance with the original cover texts than the above two methods.

2.2 Format-based Information Hiding

The format-based information hiding method is realized by changing the format of the text document. It is suit to the text with certain layout format or structure [5]. Brassil [20] hides information by adjusting its font, line space, word space, words count in one line, adding blank characters, building signature or using special formats like document head. Based on Brassil's work, Huang [21] proposes a revised approach in text watermarking where inter-word spaces of different text lines are slightly modified. Chroni [22] adopts this technology in the PDF documents. In order to enlarge the embedding rate, Mahato [23] proposes a text steganography for Microsoft Word document. The idea behind Mahato's work is that slight variation in font size of invisible character space from other characters is not reflected in the document and in the required disk size for the document. Singh [8] proposes the Text Steganography based on null spaces. Based on the big data of texts, Chen [24] proposes a coverless information hiding method using Chinese character encoding technology.

The generated documents by the format-based information hiding method are coherent to the original cover texts. However, this technique holds low embedding rare. Meanwhile, once the algorithm detail is published, it can be easily detected by the detectors. In order to enhance security, our framework employs the hash function to make the generated document more difficult to be detected. Moreover, by utilizing the hash collisions of the previously processed segments, the framework is improved to enhance the embedding rate.

3 Analysis of ASCII Characters

ASCII is a computer coding system based on the Latin alphabet, mainly for displaying of modern English and other western European languages. It is now the most common single-byte encoding system, equivalent to the international standard ISO/IEC 646. ASCII uses the combination of 7-bit or 8-bit binary numbers to represent 128 or 256 possible characters. The standard ASCII, also known as the basis ASCII, uses 7-bit binary number to represent all the English characters, Arabic numbers, punctuation marks and special control characters used in English. Table 1 lists part of invisible ASCII characters and their corresponding binary representations.

Table 1. Part of invisible ASCII characters

ASCII code	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
Definition	000 0000	000 0001	000 0010	000 0011	000 0100	000 0101	000 0110	000 0111	000 1000	000 1001

Actually, in the standard ASCII coding system, the value of codes 0 ~ 31 and 127 (33 in total) are the controlling or communicating characters (the rest are all visible characters). For example, ASCII codes like *LF* (line feed), *CR* (carriage return), *FF* (form feed), *DEL* (delete), *BS* (backspace), *BEL* (bell) are all controlling characters; ASCII codes like *SOH* (start of head), *EOT* (end of text), an *ACK* (acknowledge) are all communicating characters. The ASCII codes numbering 8, 9, 10 and 13 are converted to backspace, tabulation, line feeds, and carriage return characters respectively. These characters have no specific graphic display on the screen, but they have different effects on texts' display according to their applications.

At present, information hiding algorithms based on the substitution of invisible characters are mainly focusing on adding spaces or line breaks to some specific locations [25] or replacing blank space by null character (code as 0000000) according to secret information. For example, the famous information hiding system WbStego in the market is built based on these methods. But these methods are rarely used because of the poor robustness and relatively lower embedding rate.

After performing many different tests, we found that *SOH* (coded as 0000001) and *SP* (coded as 0100000) have similar effects in most of documents. Simultaneously, there are a lot of *SP* candidates for the replacement in the English document. So, in this paper, the *SP* and *SOH* substitution method is used to implement our information hiding framework.

4 Framework Implementation

Fig. 1 outlines the overall secret information transmission workflow of our proposed framework. Generally, the workflow can be divided into 3 stages, the information hiding stage, the information transporting stage and the secret information extracting stage. In the information hiding stage, a proper carrier document has to be prepared at first. The carrier text is not necessary a secret document and public text files (such as fiction, email, comments on news) are alright. Then, the secret information needs to be encoded into the parity code before it being embedded into the carrier document. Meanwhile, the shared secret hash key should be set between the sender and receiver. After all these works are performed, the cover text is generated for the public network transmission. In the transport stage, the cover text will be transported by the public network along with countless normal text. The interceptor seems impossible to notice such an inconspicuous cover text. In the extracting stage, on available of the shared secret hash key, the receiver can easily acquire the secret information as long as they have the same parameters configuration and the parity decoding process with the senders.

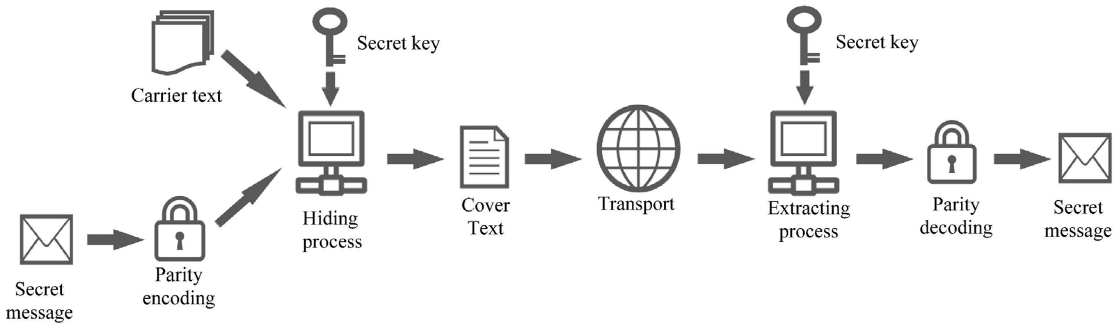


Fig. 1. The secret information transmission workflow of the framework

4.1 Sharing Setting

Before the secret message being transmitted, the shared information, such as the division of carrier document and the division of the secret information, between the sender and receiver should be set. Furthermore, the sender and receiver also need to reach an agreement on the form of hash function. Actually, some famous hash functions, such as MD4, MD5, and SHA-1, and even user defined hash functions are all feasible. In this paper, MD5 is selected as our hash function.

Carrier document division. In the secret information embedding procedure, one of the crucial work, called segmentation, has to be performed to break down the carrier document into smaller segments. On available of each smaller segment, the information hidden process is performed. However, in order to properly process the embedded secret information, the two sides have to reach an agreement on the division of the carrier document.

Actually, the simplest implementation of this division is to set a full English sentence as one segment. This method looks feasible and reasonable and it is already used in many MT-based methods. However, different division means different number of spaces in a single segment and different number of hashed values that can be compared with the present binary information. It is obviously that more spaces in a segment means high probability of embedding success. Meanwhile, the way of division of the original carrier document directly influences the embedding rate (*ER*) as Eq. (1) defines.

$$ER = \textit{(Number of bits of secret information)/(Number of bits of effective carrier document)}. \qquad \mathbf{(1)}$$

In order to reach high secret information embedding rate, it is necessary for our framework to choose a reasonable division of the carrier document. Similar to the MT-based method [15], we use the English sentence as the basic unit to divide the carrier document. But, there are mainly two differences between our method and the MT-based method: (1) the length of our segment can be freely chosen and this length is relatively fixed by setting the same *SP*s in each segment, however, the length of sentence in the MT-based method is fixed; (2) the number of *SP*s in our carrier segment is fixed and this number is under our control, however, the number of translations per sentence is undermined in the MT-based method.

Secret information division. For definiteness and without loss of generality, the secret information is encoded into the form of binary stream. Before being embedded into the segments of the carrier document, the secret information has to be split into groups with *h* bits and a process called parity coding need to be performed. By parity coding, the *h* bits of secret information will generate an *h*+1 bits parity code. Consequently, there is even number of "1" in the *h*+1 bits of parity code. As a result, there are *h*+1 bits of secret information needed to embed into one segment of the carrier document. During the process of the secret information transmission, the sender as well as the receiver needs to be in agreement on the size of *h*.

Actually, the value of *h* is generally small but it has a significant impact on the *ER*. On one hand, small *h* will result in low *ER* even if the number of hash values for a given segment is high. On the other hand, large *h* will result in frequently occurs of errors (i.e., frequently fails to find a proper hash value same as the given secret bits). Supposes there are *k SP*s in a given segment of the carrier document, the failure probability (*PR*) with a given number *h* can be calculated as the formula below:

$$PR=(1-(1/2)^{h+1})^{k}. \qquad \mathbf{(2)}$$

In order to acquire high *ER*, relationship between the value of *h* and *k* needs to be carefully considered. By analyzing the experimental results (as described in Sect. 6), we set $h\in[1, 7]$ and $k\in[2^3, 2^6]$ in our framework. However, even if we get the reasonable value of *h* and *k*, it is also possible that there is no hash values for a given carrier segment and a group of secret information. The occurrence of this situation is defined as *PNH* (possibility of no hash value) as Eq. (3) describes.

$$PHN = \begin{cases} (1/2)^{k}, h+1 \text{ is odd} \\ (1-(1/2)^{h+1}-1/2)^{k}, h+1 \text{ is even.} \end{cases} \qquad \mathbf{(3)}$$

From Eq. (3), we can see that the *PNH* is relatively small. In our improved method (in the upcoming Sect. 5), this value can be make so smaller to be negligible.

Shared secret key. The sender and receiver have to set a shared secret key for the hash process. As the secret key and the secret information are transmitted as the same way, the interceptors do not aware the existence of the secret shared key. So, they cannot detect the existence or even extract the secret message by the hash process. If the secret key alone is considered to be not strong enough, the hidden message itself can additionally be encrypted with another secret key prior to the steganographic encoding process.

4.2 Embedding Procedure

On available of parity coded binary secret stream, the following three steps are employed to embed the secret information into the carrier document.

Secret information encoding. Firstly, it is supposed that the length of the secret information is L bits and it been split into groups with h bits. Then, a parity bit is added to ensure that there is even number of "1" in each group. As a result, there are h+1 bits in each group for the secret information embedding procedure. However, as processed by Fig. 2, if L is not an integer multiple of h, the last group will be automatic completed. Actually, the additional bits will be neglected during the secret information extraction procedure. So, it is not necessary for the sender and receiver to have an agreement on the additional bits. They just need to ensure that there is even number "1" in the last group.

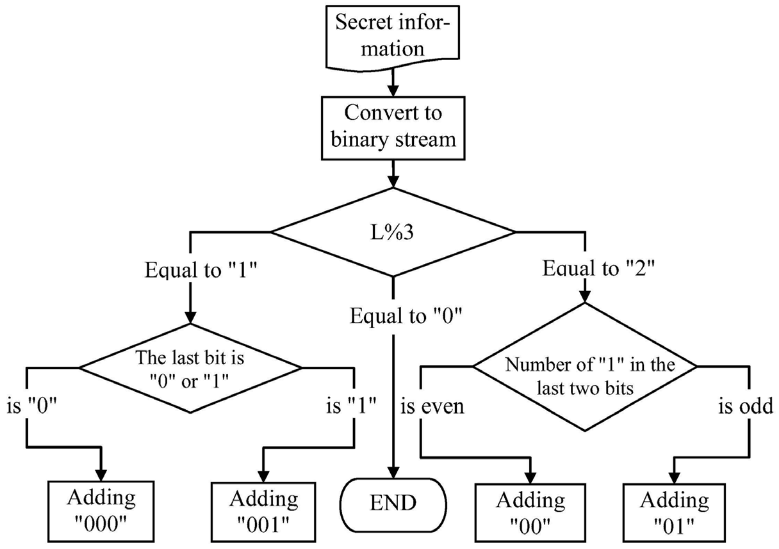


Fig. 2. An example of secret information encoding process (h=3)

Information substitution. Firstly, it has to choose a text-based document as the carrier document. Then, the segments in the carrier document are fetched in turn. Each segment is processed by the substitution function to replace the *SP* with the *SOH*. For security reason, only one *SP* is selected from a segment for substitution. Although, replaces more than one spaces will bring more options. However, more options will results in more changes on the original document. Consequently, more changes will attract more attentions from the interceptors.

Replaced segment hashing. Each replaced segment is firstly hashed with a secret hash key. Then, the hash values will be compared with the encoded secret parity group composed of h+1 bits. During this process, if there is more than one hash values equal to the secret message, we can randomly select one. However, if there is no result same as the secret message, results that have odd number of "1" can be randomly selected. It has to notice that the selected odd "1" result contains no secret information.

Repeats the above steps until all the secret information is successfully embedded into the carrier document. Latterly, copies the rest part of the original carrier document into the newly generated document to complete to embedding procedure.

4.3 Extracting Procedure

Compared with the embedding procedure, the extracting procedure is relatively simple as long as the receiver has the same configuration with the sender. The workflow of the secret information extraction procedure can be described as follows:

(1) Orderly applies the same hash function and the same secret hash key on each segment of the received English document.

(2) If the first h+1 bits of a derived hash value has even number of "1", then the first h bits of this value are the secret message.

(3) Otherwise, if the derived hash value has odd number of "1", then this hash value can be neglected since it has no secret information.

(4) Repeatedly reads the segments from the received document until a segment with no *SOH* character is met.

As described in Fig. 2, there might be some additional bits in the last segment of the document. Since these bits do not affect the secret information transmission, they don't need any special operations.

5 Framework Improvement

In our framework, the hash function is employed to make the secret information that been embedded into the carrier document stronger to be detected. However, the phenomenon of collisions, namely different initial values may have the same hash result, is inevitable. In this section, we will propose an improved method which can settle the collisions effectively.

In our method, collision happens when the hash values derive from a single English segment happen to the same results in their highest h+1 bits. Specifically, for an original segment A, X_1 is a segment generated by replacing the first SP of A; X_2 is another segment generated by replacing the second SP of A. $Hash_1$ is the hash value comes from X_1, and $hash_2$ (processed by the same hash function as X_1) comes from X_2. Under this situation, if the highest h+1 bits of $hash_1$ and $hash_2$ are identical, we can say that the collision is happened, and the X_1 and X_2 are called collision replacements. The existence of collision means less choice of candidate hash values.

According to the "birthday paradox" theory, when there are k SPs in a segment, the probability of not equal happens in the highest h+1 bits of any two hash values that are derived from this segment can be calculated as $\Pi_{i=1}^{k-1}(1-i\times 2^{-(h+1)})$, where k<2(h+1). However, if $k \geqq 2(h+1)$, this probability is 0. In most cases, this probability (not equal) is pretty small even if we get the reasonable h and k. So, the collision frequently happens during the hashing process which is described in Sect. 4.

In this section, we will propose an improved method which can settle the collisions effectively. In this improvement, the existence of hash collisions of a former segment can be used to help reducing the occurrence of collisions of its subsequent segments by providing them with a richer set of available hash values.

In order to better illustrate the improved method, we define that there are two cases of collision replacements, the even collision and the odd collision. The even collision happens when several replaced segments derived from a single original segment hash to values with the same part of their highest h+1 bits. For all the hash values of the replaced segments that are derived from a carrier segment, if they are different from the secret information needs to be hidden, then the hash values with odd number of "1" are called odd collision. The even collisions and odd collisions consist of all collisions of a carrier document. In our improved method, collisions of the former segments can be used to generate more hash values to its subsequent segments.

For the purpose of effectively utilizing the existed collisions, a window W with C consecutive English segment is set. In this window, the numbers of the hash values come from the i^{th} segment are influenced by the collision replacements of all the segments prior to i. In the hash procedure, the current replaced segment will be hashed accompany with one of the former collisions. Namely, if p and q represent the current replaced segment and one of the former collision replacement respectively, then $hash(p+q)$ is the hash value of p.

If the $(i\text{-}1)^{th}$ segment has N collision replacements, the number of available hash values of i^{th} segment will hike up N times. The growing number of hash values directly led to the probability of the existence of hash values same as the given secret bits for the i^{th} segment. Specifically, the amount of hash values for segment i (AH) can be calculated as:

$$AH = K_i \times \prod_{j\text{-}1}^{i\text{-}1} N_j \tag{4}$$

In Eq. (4), K_i represents the amount of SPs in the $i^{th}(0<i<C)$ segment and N_j represents the amount of the collision replacements of the $j^{th}(0<j<i)$ segment. From the equation, we can see that AH grows with C exponentially. A relatively bigger C will make AH an unacceptable huge number. So, it is needed to set an upper bound t $(AH \leq t)$ for AH. The reasonable value of t is affected by the hardware environment as well as the value of h and k. Here, h and k refer to the number of bits in a secret information group and the number of SPs in a segment of carrier document respectively.

In our implementation, a window that accommodates two consecutive segments is set, namely $W=2$. From the above analysis, a larger W could bring more available hash values. But larger W also costs more computing resources and more space to buffer hash values. Furthermore, the threshold of the number of hash values for a specific segment is set as t. The detailed embedding procedure of our improved method is described as follows:

(1) Encodes the initial secret information to ensure that there is even number of "1" in each group (same as the first step of Sect. 4.2).

(2) For each window W, utilizes ASCII character *SOH* to replace the i^{th} $(0 < i \leq k)$ *SP* of the first segment to get a k elements set X ($X=\{X_i \mid X_i$ is the replaced segment $\otimes$ $0<i\leq k\}$); Then, applies hash function on these replaced segments to get k hash values; At last, compares all these hash values with the given secret bits to get a collision set P ($P = \{P_i \mid P_i$ is an even or odd collision $\otimes$ $0<i\leq k<t\}$).

(3) Orderly selects $P_i \in P$, $Y_j \in Y$ ($Y=\{Y_j \mid Y_i$ is a replaced segment generated from the 2nd segment of W $\otimes$ $0 < j \leq k\}$), and utilizes the hash function $hash(P_i+Y_j)$ to get a new group of hash values. Here, the number of new generated hash values is k times with the number of elements of P. Then, compares this group of hash values with the given secret bits to get a replaced collision set Q ($Q =\{Q_i \mid Q_i$ is an even or odd collision $\otimes$ $0<i\leq k\}$) of the second segment of W. Q is a subset of $Y_i \times P_i$. By now, the embedding procedure of the first window is finished.

(4) By the previous steps, two replaced collision sets, P and Q, may be generated for embedding the secret information. If P is generated, since it must contain some even collisions, we can randomly select an even collision as the carrier segment. Otherwise, only Q is generated:

(a) if there is an even collision q in Q and q is derived from an even collision p in P (which means the result of $hash(p+q)$ is equal to the given secret bits), we can embed $2h$ secret bits (not including the parity bit) in the present windows W. However, if q is derived from an odd collision of P, we can only embed h secret bits.

(b) If all elements in Q are odd collisions, corresponding to the two cases of (a), only h bits or 0 bits are able to be embedded.

By considering all possible situations, proper segments of the carrier document are selected to embed the maximum number of secret bits.

(5) If the all the secret bits are embedded, terminates the embedding procedure. Otherwise, continues processing the next window, turn to Step (2).

6 Experimental Evaluation

This section evaluates the performance of our framework from perspectives of embedding rate, robustness and efficiency respectively. The experiments in this section are carried out on the machine with Intel i7 4790 CPU (3.6 GHz), 8GB DDR3 1600 RAM and 64-bits Windows 7 OS. The carrier documents are .doc and .txt files written in English.

6.1 Embedding rate analysis

Theoretical maximum embedding rate. *MER* (maximum embedding rate) is firstly mentioned in the machine translation based (MT-based) information hiding method [15]. In MT-based method, *MER* means the ratio of the theoretical maximum secret bits that can be embedded into an English sentence to the average length of the English sentences. The ratio is influenced by the quantity of the translations per sentence. More translations mean larger *MER*. The MT-based method uses the English sentence as the basic embedding unit. However, diverse lengths of sentences and different number of translations of a specific sentence directly lead to poor *MER*. In order to facilitate processing, they statistically set 1168 bits as the average length of a sentence. Some *MER* of MT-based method is shown in Table 2 (*k'* represents the average number of translations per sentence; *AL* represents the average length of English sentences).

Table 2. *MER* of MT-based method

k'	11.62	15.15	18.01
AL	1168	1168	1168
MER(%)	0.22	0.27	0.31

In this paper, we set the English segments correspond to the English sentences of the MT-based method, the number of *SP*s in a segment correspond to the number of translations per sentence of the MT-based method. There are two differences between our method and the MT-based method: (1) the length of our segment can be freely chosen and this length is relatively fixed by setting the same *SP*s in each segment, however, the length of sentence in the MT-based method is fixed; (2) the number of *SP*s in our segment is fixed and this number is under our control, however, the number of translations per sentence is undermined in the MT-based method.

According to the definition of *ER* described in Sect. 4.1, the *MER* of our method can be defined as:

$$MER = \log k / \mathrm{L}_k. \quad \textbf{(5)}$$

In this definition, L_k represents the length (we use bit as unit) of the segment with k spaces. Table 3 shows the *MER* of our method under different value of k (*AL* represents the average length of our segment under different value of k).

Table 3. *MER* of our method

k	2	4	8	16	32	64
AL	11.6	23.6	47.4	95.0	190.6	381.2
MER (%)	1.14	1.09	0.80	0.53	0.33	0.20

In Table 2, *MER* is increasing with the growing of k'. However, results shown in Table 3 are the opposite: *MER* is decreasing with the growing of k. The value of k' in Table 2 is affected by the underlying translation machine. In order to ensure the translation quality, too large k' is not acceptable. As a result, the MT-based method has a relatively small *MER* (the maximum *MER* is about 0.33% in practice). Since the value of k is changeable, our method can reach an ideal *ER* when a proper k is chosen. In Table 3, *MER* reaches large value when k is set as 2, 4 or 8. However, too large *MER* may lead to the failure of the information embedding procedure. As a compromise, we choose 8 as the proper value of k.

The effect of h and k on ER. The above section analyzed the *MER* of our method theoretically. But in reality, the embedding error and the redundant bits introduced by parity code make the procedure cannot reach such high *ER*. As a matter of fact, we have a much lower *ER* in our original method. However, we are able to approach the *MER* when the reasonable values of h and k are chosen as our improved method does. In our methods, k represents the number of *SP*s in a carrier segment; h records the number of bits in a secret information group. Fig. 3 and Fig. 4 depict the effects of h and k on *ER* before and after the improvement. Since it is often not operable for the value of *x-axis* less than 4, the value of *y-axis* is basically 0 when the value of *x-axis* is less than 4.

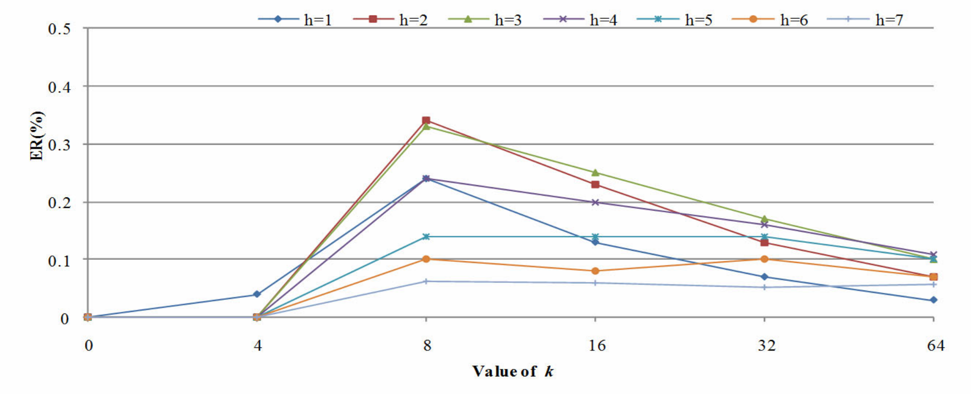


Fig. 3. The effect of h and k on *ER* of our original method

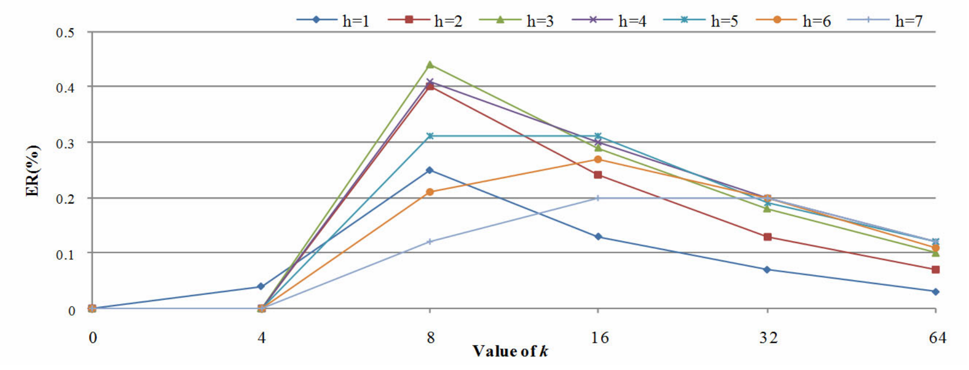


Fig. 4. The effect of *h* and k on *ER* of our improved method

The two figures also show that *ER* reaches a relatively large value when *k* is set as 8 or 16 and *h* is set as 2, 3 or 4. Obviously, *ER* has a considerable increment after the improvement. In the improved method, the largest two values of *ER*, 0.44% and 0.41%, are achieved by setting *k* as 8 and 16 respectively. It is improved more than 30% compared with the original method.

However, there is still a big gap between the largest *ER* in Fig. 4 and the *MER* in Table 3. Fig. 3 and Fig. 4 also show that *ER* is not regularly increasing or decreasing with the increment of *h* and *k*. As a matter of fact, the increment of *k* and *h* will result in larger capacity of the carrier segments; larger capacity of each carrier document will result in longer carrier segment; longer segment will result in the higher probability of embedding error since it will bring a lot of carrier segments with no embedded secret information.

The effect of h and k on the embedding error. As mentioned previously, *ER* is not regularly increasing or decreasing with the increment of *h* and *k*. In order to get the best *ER*, we need to evaluate the influences of *h* and *k* on the occurrence of embedding errors. Fig. 5 and Fig. 6 give the results of embedding errors with different *h* and *k* before and after the improvement.

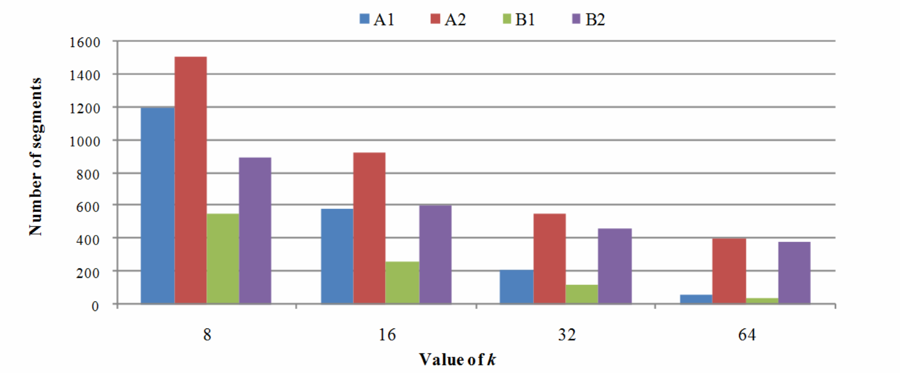


Fig. 5. “Empty” with different *k* (*h*=4)

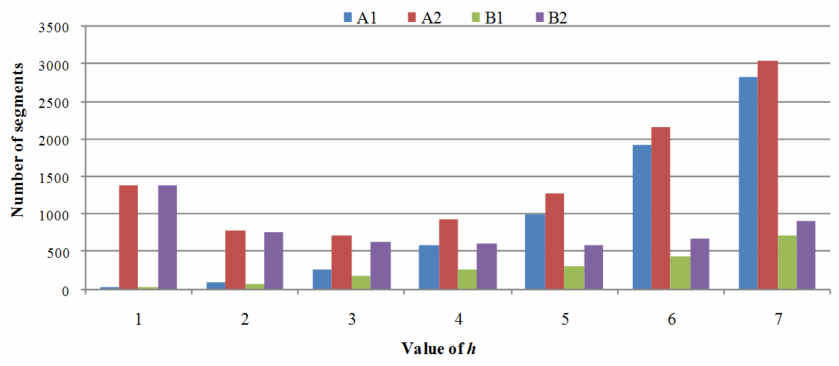


Fig. 6. “Empty” with different *h* (*k* =24)

In Fig. 5, *A1* and *A2* represent the number of carrier segments with no embedded secret information and the total number of segments in carrier document before improvement respectively. *B1* and *B2* represent the same results after the improvement. In this experiment, the length of secret information is 1376 bits (1720 bits after parity encoding). The secret information are divided into groups with 4 bits (i.e. *h*=4). From the figure, we can see that all the results, *A1*, *A2*, *B1* and *B2*, are decreasing with the growing number of *k*. *B1* and *B2* decrease faster than *A1* and *A2*. In Fig. 6, *A1*, *A2*, *B1* and *B2* represent the same values as the ones in Fig. 5. In this experiment, the number of *SP*s in each carrier segment is set as 16 (i.e. *k*=16). From this figure, we can see that *A2* and *B2* are decreasing with the growing of *h* until it arrives at 3. This is because smaller *h* will get relatively lower rate of embedding error. As shown in Fig. 5, *A1* and *B1* are relatively small compared with *A2* and *B2*. So, *A2* and *B2* are almost only affected by the capacity of the carrier segments. However, *A1* and *B1* are increasing rapidly with the growth of *h* when it exceeds 3. Under this circumstance, *A1* and *B1* take bigger and bigger proportions of *A2* and *B2* respectively. As a result, *A2* and *B2* also increase rapidly with *h*. Fig. 6 also shows that *B1* and *B2* are much smaller than *A1* and *A2* respectively. Which means our improved method gets better *ER* than the original method.

Although we get better *ER* than the original method, there is still a gap between our best *ER* and theoretically *MER*. By performing further experiments, we have found that the first segment in the carrier windows is the biggest factor for the occurrence of the embedding error. As a matter of fact, our experiments only place two segments to each carrier windows (i.e. *W*=2) for simplicity. The improved method only increased the hash values to the second segment of the carrier window. But the first segment is processed as the original method. For this reason, we can place more segments to the carrier windows to get higher *ER*.

6.2 Robustness Analysis

The robustness of our framework can be carried out by accessing its stability and security respectively.

Stability analysis. As mentioned previously, the embedding failure may occur during the information hiding stage. In our framework, the embedding failure occurs when there is no hash values same as the secret information to be hidden or there is no hash value contains even number of “1”. We have to find a new carrier document when the embedding failure occurs. The high occurrences of embedding failure will reduce the stability and feasibility of the information hiding algorithm. Theoretically, if two bits secret information (*h*=2) are embedded into carrier segments with 8 *SP*s (*k*=8), we will get 99.961% embedding success rate for each carrier segment, 99.6% embedding success rate for 10 consecutive carrier segments, and 96.2% embedding success rate for 100 consecutive carrier segments.

Table 4 lists the embedding success rate of our improved method by testing documents with different values of *h* and *k*. In this experiment, the parameter *W* is set as 2. The results in the table are the average results by deploying 100 experiments with different carrier documents and different fixed-length secret information. In the table, *CNum* represents the number of secret characters needed to be embedded into the carrier document.

Table 4. The success embedding rate of our improved method (%)

h, *k*	*CNum*						
	5	10	20	50	100	150	200
h=2, *k*=8	100	100	100	75	63	61	49
h=2, *k*=16	100	100	100	100	100	100	100
h=3, *k*=8	98	97	96	87	91	79	82
h=3, *k*=16	100	100	100	100	100	100	100
h=4, *k*=8	97	98	99	86	66	59	57
h=4, *k*=16	100	100	100	100	100	100	100

From the table, we can see that our improved method can reach more than 90% embedding success rate in most cases. However, we also have low embedding success rate in some cases. But these situations are still operational in practice. Furthermore, we can avoid the occurrence of these situations by adjusting the corresponding parameters. As a whole, we have achieved a relatively high embedding success rate, namely higher stability.

Security analysis. There are two criteria for evaluating the security of the information hiding algorithms:

(1) the ability of covering the "noises (i.e. changes)" that the hiding algorithms bring into the original carrier documents cannot be easily detected by the interceptors; (2) the ability of preventing the secret message being extracted by the interceptors when they are aware of the existence of this message.

(1) *Ability of Hiding*

As mentioned in Sect. 3, by performing many different tests, we found that the *SP* and *SOH* have the similar effects in most of documents. Meanwhile, there are many candidate *SP*s in the document for replacing, our framework uses *SP* and *SOH* substitution method to implement the information hiding algorithm. However, similar do not means the same. In the document, the width of *SP* is shorter than the ones of *SOH*. This means that there are many abnormal gaps with different widths among words of the document that carrying secret messages. This kind of "noise" may not be able to resist the test of detection algorithm based on word shift. In order to cope with these detection algorithms, we introduce the word shift based hiding method to our algorithm to make *SOH* has the absolutely same width with *SP*.

Another "noise" our framework brings into the original carrier document may be the probability of *SP* (denoted as r_1) and the probability of continuous *SP* (denoted as r_2). In the generated document that been embedded secret message, r_1 is defined as the number of all *SP*s divides by the number of all characters; r_2 is defined as the number of continuous *SP*s divides by the number of all *SPs*. Theoretically, the threshold for r_1 is 0.3 and the threshold for r_2 is 0.2 [26]. Since we put the restriction on the value of *k*, the values of r_1 and r_2 in our method are much smaller than the theoretically ones.

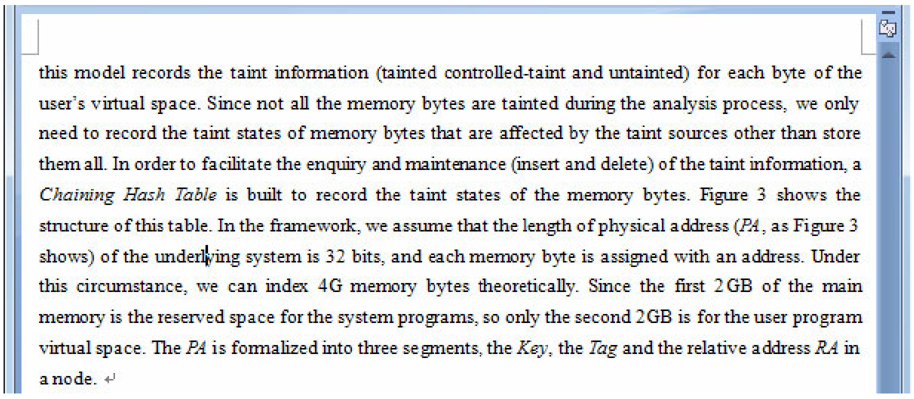
Table 5 lists the average results of our experiments by testing 100 different documents. In this table, the first line (*Original*) lists the results of the original carrier documents. The rest lines are all the results of the generated documents that been embedded with secret message with different *h* and *k*.

Table 5. Probability of *SP* and Probability of continuous *SP*

h, *k*	r_i	
	r_1	r_2
Original	0.1893	0.0138
h=2, *k*=8	0.1630	0.0086
h=2, *k*=16	0.1767	0.0106
h=3, *k*=8	0.1628	0.0084
h=3, *k*=16	0.1464	0.0090
h=4, *k*=8	0.1643	0.0076
h=4, *k*=16	0.1764	0.0089

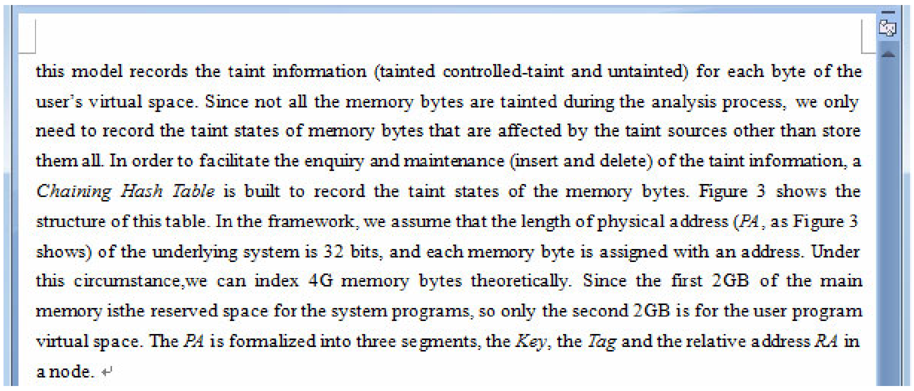
From the Table 5, we can see that results of our method are much better than the threshold ones in [15]. The generated documents of our method are totally met the demands of the normal document from perspectives of semantics, syntaxes and statistic characteristics respectively. As a result, detection algorithms, such as natural language processing based method, different form synonym replacement based method and MT-based method, cannot work well on our algorithm.

Fig. 7 and Fig. 8 give the secret information hiding effect of our framework by embedding secret information into the .doc document. In our improved method, the inputs of the embedding procedure are the carrier document and the secret message. The output is the "normal" document that that been embedded with the secret message. Fig.7 displays a piece of the original carrier document. Fig.8 is the corresponding document that been embedded with secret information ("Brussels followed Washington's lead in 1989 by a company of seven."). From the two figures, we can see that the differences that brought by the embedding procedure brings can be neglected except for the format distinctions that built-in different types of files. Furthermore, since our framework is built based on the code structure of the original carrier document, the changes of the font and size of the document have no influences on the embedded secret information.



this model records the taint information (tainted controlled-taint and untainted) for each byte of the user's virtual space. Since not all the memory bytes are tainted during the analysis process, we only need to record the taint states of memory bytes that are affected by the taint sources other than store them all. In order to facilitate the enquiry and maintenance (insert and delete) of the taint information, a *Chaining Hash Table* is built to record the taint states of the memory bytes. Figure 3 shows the structure of this table. In the framework, we assume that the length of physical address (*PA*, as Figure 3 shows) of the underlying system is 32 bits, and each memory byte is assigned with an address. Under this circumstance, we can index 4G memory bytes theoretically. Since the first 2GB of the main memory is the reserved space for the system programs, so only the second 2GB is for the user program virtual space. The *PA* is formalized into three segments, the *Key*, the *Tag* and the relative address *RA* in a node. ↵

Fig. 7. Original carrier document (.doc)



this model records the taint information (tainted controlled-taint and untainted) for each byte of the user's virtual space. Since not all the memory bytes are tainted during the analysis process, we only need to record the taint states of memory bytes that are affected by the taint sources other than store them all. In order to facilitate the enquiry and maintenance (insert and delete) of the taint information, a *Chaining Hash Table* is built to record the taint states of the memory bytes. Figure 3 shows the structure of this table. In the framework, we assume that the length of physical address (*PA*, as Figure 3 shows) of the underlying system is 32 bits, and each memory byte is assigned with an address. Under this circumstance,we can index 4G memory bytes theoretically. Since the first 2GB of the main memory isthe reserved space for the system programs, so only the second 2GB is for the user program virtual space. The *PA* is formalized into three segments, the *Key*, the *Tag* and the relative address *RA* in a node. ↵

Fig. 8. Document embedded with secret message (.doc)

Ability of resisting attack. Unfortunately, if a smart interceptor happened to smell the existence of the secret message, he would find it is a significantly hard procedure to extract the secret message without known about the embedding algorithm and the values of relative parameters of our method. At present, there is no effective algorithm in the market has the ability of extracting the secret message from all the information hiding systems. Beside the values of *h*, *k*, *w*, *t* and the location of the bits in hash values that used to compare with the given secret bits, we also employ a secret key in hash function. Brute force the hash function, like MD5, SHA-1, SHA-2, is proved harder. In addition, we can also construct a unique hash function and encrypt the secret messages beforehand to enhance the security of our system.

6.3 Efficiency Analysis

The traditional information hiding technology has no strict restriction on the efficiency. An information hiding algorithm is acceptable as long as it has high robustness and high embedding rate. However, with the rapid development of mobile communication environment, it is urgently needs "good" (with relatively lower runtime overhead) information hiding algorithms that can be deployed on the resource starvation mobile platforms. Based on this observation, this subsection evaluates the efficiency of the proposed framework.

Table 6 and Table 7 list the experimental results by evaluating the performance our framework with different parameters (*W*=2). Table 6 lists the time cost of the information hiding stage. Table 7 lists the time cost of the information extracting stage. Results of these tables are the average data by deploying 100 experiments with different carrier documents and different fixed-length secret information on our framework. In the two tables, *CNum* represents the number of secret characters needed to be embedded into the carrier document.

Table 6. Time overhead of the secret information hiding stage (*ms*)

h, *k*	*CNum*						
	5	10	20	50	100	150	200
h=2, *k*=8	21.2	29.9	44.6	77.3	109.8	127.9	146.0
h=2, *k*=16	33.4	49.0	78.1	112.2	150.5	127.7	203.8
h=3, *k*=8	25.0	31.5	46.2	77.9	110.0	125.6	141.2
h=3, *k*=16	37.7	50.4	81.8	117.1	157.8	179.6	205.8
h=4, *k*=8	25.8	34.6	49.8	89.8	116.5	129.3	148.5
h=4, *k*=16	46.2	64.0	91.3	126.8	167.3	203.2	227.9

Table 7. Time overhead of the secret information extracting stage (*ms*)

h, *k*	*CNum*						
	5	10	20	50	100	150	200
h=2, *k*=8	12.6	13.8	17.0	26.1	37.4	46.0	54.1
h=2, *k*=16	12.0	15.0	18.8	27.4	35.8	44.9	52.4
h=3, *k*=8	13.2	13.9	14.5	26.2	35.6	44.6	53.0
h=3, *k*=16	12.7	13.4	16.8	24.7	33.2	39.5	48.9
h=4, *k*=8	13.0	14.1	18.2	26.6	35.7	43.0	53.3
h=4, *k*=16	12.3	13.2	16.7	23.4	32.0	41.8	45.8

From Table 6, we can see that the runtime overhead of the information hiding stage is directly proportional to the number of characters of the secret information. Since larger value of *k* will directly lead to the large number of computations of hash function, this overhead also grows with the increasing value of *k*. The value of *h* also has slight influence on the runtime overhead of the information stage. Similarly, as Table 7 lists, the runtime overhead of the information extracting stage is also directly proportional to the number of characters of the secret information. However, both *h* and *k* have slight influence on the runtime overhead of the information stage. From this experiment, we can see that the overhead of our framework is relatively low, and it is applicable to the mobile platforms.

7 Conclusion and Future Work

As an important topic in the information security, the current text-based information hiding techniques are suffering from poor robustness, lower embedding rate and semantic clutter. In order to solve these problems, this paper proposed a novel secret information hiding framework based on the integration of hash function and the invisible ASCII character replacement technology. The framework was tested from perspectives of embedding rate, robustness and efficiency respectively. The experimental results have demonstrated that the proposed framework holds high embedding rate, strong robustness and high efficiency. However, only applicable for the English documents that have plenty of space symbols seriously restricts the applications of the proposed algorithm. So, in the future, it is urgently needed to apply this algorithm to a wider range of languages.

Acknowledgments

This work is partly supported by the academic and technical leader recruiting policy of Anhui University, the National Natural Science Foundation of China (Grant Nos.61300169, 61702004), and the Natural Science Foundation of Education Department of Anhui province. (Grant Nos. KJ2016A257).

Reference

[1] P. Moulin, J.A. O'Sullivan, Information-theoretic analysis of information hiding, IEEE Transactions on Information Theory 49(3)(2003) 563-593.

[2] M. Köhler, I. Lukić, V.K. Čik, Protecting information with subcodstanography, Security & Communication Networks

2017(1)(2017) 1-13.

[3] C.-Y. Yang, Robust high-capacity watermarking scheme based on Euclidean norms and quick coefficient alignment, Multimedia Tools and Applications 76(1)(2017) 1455-1477.

[4] A.S. Lorente, S. Berres, A secure steganographic algorithm based on frequency domain for the transmission of hidden information, Security & Communication Networks 2017(2)(2017) 1-14.

[5] Y.-B. Luo, Y.-F. Huang, F.-F. Li, C.-C. Chang, Text steganography based on Ci-poetry generation using Markov chain model, KSII Transactions on Internet and Information Systems 10(9)(2016) 4568-4584.

[6] A.V. Subramanyam, S. Emmanuel, M.S. Kankanhalli, Robust watermarking of compressed and encrypted HPEG-2000 images, IEEE Transactions on Multimedia 14(3)(2012) 703-716.

[7] M. Botta, D. Cavagnino, V. Pomponiu, Protecting the content integrity of digital imagery with fidelity preservation: an improved version, ACM Transactions on Multimedia Computing, Communications, and Applications 10(3)(2014) 96-105.

[8] P. Singh, R. Chaudhary, A. Agarwal, A novel approach of text steganography based on null spaces, IOSR Journal of Computer Engineering 3(4)(2012) 11-17.

[9] M.S. Rahman, I. Khalil, X. Yi, H. Dong, Highly imperceptible and reversible text steganography using invisible character based codeword, in: Proc. the Twenty First Pacific Asia Conference on Information Systems, 2017.

[10] F. Liu, P.-P. Luo, Z.-J. Ma, C. Zhang, Y. Zhang, E. Zhu, Security secret information hiding based on hash function and invisible ASCII characters replacement, in: Proc. the 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2016.

[11] K. Maher, TEXTO. <http://www.nic.funet.fi/pub/crypt/steganography/>, 1995 (accessed 18.01.06).

[12] P. Wayner, Disappearing Cryptography: Information Hiding: Steganography and Watermarking, 2nd ed., Morgan Kaufmann, San Francisco, 2002.

[13] M. Chapman, G.I. Davida, M. Rennhard, A practical and effective approach to large-scale automated linguistic steganography, in: Proc. the 4th International Conference on Information Security, 2001.

[14] L.-Y. Xiang, X.-H. Wang, C.-F. Yang, P. Liu, A novel linguistic steganography based on synonym run-length encoding, IEICE Transactions on Information & Systems E100.D(2)(2017) 313-322.

[15] R. Stutsman, C. Grothoff, M. Atallah, K. Grothoff, Lost in just the translation, in: Proc. the 2006 ACM Symposium on Applied Computing, 2006.

[16] P. Meng, L.-S. Hang, Z.-L Chen, Y.-c. Hu, W. Yang, STBS: a statistical algorithm for steganalysis of translation-based steganography, in: Proc. the 12th International Conference Information Hiding, 2010.

[17] C. Grothoff, K. Grothoff, R. Stutsman, L. Alkhutova, M. Atallah, Translation-based steganography, Journal of Computer Security 17(3)(2009) 269-303.

[18] E. Satir, H. Isik, A Huffman compression based text steganography method, Multimedia Tools and Applications 70(3)(2014) 2085-2110.

[19] L.-Y. Xiang, G. Luo, Y.-H. Xie, W. Chen, Research on the coding strategies for synonym substitution-based steganograph, Journal of Computational Information Systems 10(22)(2014) 9877-9888.

[20] J.T. Brassil, S. Low, N.F. Maxemchuk, Copyright protection electronic distribution of text documents, in: Proc. the IEEE 87(7)(1999) 1181-1196.

[21] D. Huang, H. Yan, Interword distance changes represented by sine waves for watermarking text images, IEEE Transactions on Circuits and Systems for Video Technology 11(12)(2001) 1237-1245.

[22] M. Chroni, S.D. Nikolopoulos, Watermarking PDF documents using various representations of self-inverting permutations, in: Proc. the 11th International Conference on Web Information Systems and Technologies, 2015.

[23] S. Mahato, D.K. Yadav, D.A. Khan, A novel approach to text steganography using font size of invisible space characters in Microsoft word document, Intelligent Computing, Networking, and Informatics 243(2014) 1047-1054.

[24] X.-Y. Chen, H.-Y. Sun, Y. Tobe, Z. Zhou, X. Sun, Coverless information hiding method based on the Chinese mathematical expression, in: Proc. 1st International Conference on Cloud Computing & Security, 2015.

[25] A.A. Mohamed, An improved algorithm for information hiding based on features of Arabic text: a unicode approach, Egyptian Informatics Journal 15(2)(2014) 79-87.

[26] X.-G Sui, H. Luo, A steganalysis method based on the distribution of space characters, in: Proc. 4th International Conference on Communications, Circuits and Systems, 2006.