New Text Steganography Technique Based on Part-of-Speech Tagging and Format-Preserving Encryption

Mohammed Abdul Majeed[1*], Rossilawati Sulaiman[1], and Zarina Shukur[1]
[1] Center for Cyber Security, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia,
43600 Bangi, Selangor, Malaysia
[e-mail: p97938@siswa.ukm.edu.my, rossilawati@ukm.edu.my, zarinashukur@ukm.edu.my]
[*]Corresponding author: Mohammed Abdul Majeed

Received February 26, 2023; accepted November 26, 2023; published January 31, 2024

Abstract

The transmission of confidential data using cover media is called steganography. The three requirements of any effective steganography system are high embedding capacity, security, and imperceptibility. The text file's structure, which makes syntax and grammar more visually obvious than in other media, contributes to its poor imperceptibility. Text steganography is regarded as the most challenging carrier to hide secret data because of its insufficient redundant data compared to other digital objects. Unicode characters, especially non-printing or invisible, are employed for hiding data by mapping a specific amount of secret data bits in each character and inserting the character into cover text spaces. These characters are known with limited spaces to embed secret data. Current studies that used Unicode characters in text steganography focused on increasing the data hiding capacity with insufficient redundant data in a text file. A sequential embedding pattern is often selected and included in all available positions in the cover text. This embedding pattern negatively affects the text steganography system's imperceptibility and security. Thus, this study attempts to solve these limitations using the Part-of-speech (POS) tagging technique combined with the randomization concept in data hiding. Combining these two techniques allows inserting the Unicode characters in randomized patterns with specific positions in the cover text to increase data hiding capacity with minimum effects on imperceptibility and security. Format-preserving encryption (FPE) is also used to encrypt a secret message without changing its size before the embedding processes. By comparing the proposed technique to already existing ones, the results demonstrate that it fulfils the cover file's capacity, imperceptibility, and security requirements.

Keywords: Text Steganography, Format-based Steganography, Part of Speech, Format-Preserving Encryption, Unicode characters.

http://doi.org/10.3837/tiis.2024.01.010 **ISSN : 1976-7277**

1. Introduction

Steganography name contains two ancient Greek words: "Stegano" and "Graphy", and both relate to "Cover Writing". Steganography has been in use for decades [1]. As compared to other media like images, sounds, and video, text contains fewer redundant bits [2]. The secret data embedding in text media is the main concern in text steganography. Little modifications to the cover text will be visible as redundant bits are scarce [3]. Three characteristics are required in any steganography system: security, imperceptibility, and capacity [4][5]. A steganography system often gets to transmit hidden data via the least-covered media, which lessens the possibility of detected while transmitting through an unsafe channel. Therefore, storing secret data requires high embedding capacity [6][7]. These characteristics have the most influence on how effective a steganography setup is. However, there is generally a trade-off between the secret data capacity and the stego file quality [8][9]. For example, embedded a large quantity of secret data. In such cases, the stego files changing presents extra difficulties in terms of maintaining imperceptibility and the chance for distortion. Therefore, the primary goal of any successful proposed method is to maintain these characteristics as effectively as possible [10].

Text steganography can be divided into three classes: format-based, random and statistical generation, and linguistic [11],[13]. Firstly, in the format-based method, Text symbols' physical characteristics are employed. The changed characteristics are performed so that the human eye cannot recognize the changes [14]. In encoding the format-based method, the physical attributes of the words are changed to hide the secret data, which is dependent on symbols in different languages. Various studies focus on improving text steganography's capacity by changing the text format's physical nature using format-based approaches [15].

Secondly, in the random and statistical generation method to create cover texts, statistical features of a language are acquired. In each language, sequences of characters in a text and statistical properties, including word lengths or the frequency of occurrences of letters, are utilized in text documents [16].

Lastly, language characteristics hide secret information inside text files in the linguistic approach [17]. For example, POS can be used in English to analyze cover and secret texts and can list the definition for the text content in terms of nouns, verbs, adjectives, etc. [18]. POS is used to replace each word with a similar tag (noun to noun, verb to verb, etc.) between the secret and cover text. Alternatively, deep learning (DL) can provide an auto-generation of text used in steganography to auto-generate a cover text that already carries the secret data [19]. Consequently, the security of the cover text can be enhanced by minimizing the cover text's and stego text's visual appearance after embedding secret bits into the cover text [20]. However, Linguistic steganography raises a new challenge in security because of the semantic control of the generated texts, which are detectable by attackers.

Previous work in [9] reviewed techniques used in text steganography. One of the main issues with the embedding techniques that used non-printing or invisible Unicode characters in format-based text steganography was the position selections for embedding secret data in the cover text. Cover texts are known with limited spaces that can be used for embedding secret data, which also applies to Unicode characters. Despite these proposed approaches according to a specific language, they share a common way of selecting embedding positions inside the text: a sequential selection pattern that includes all available positions in the text document. In addition, sequentially embedding secret data in a text document where human eyes can easily detect changes negatively affects the imperceptibility and security. It makes the algorithm vulnerable to detection by a third party [21]. Therefore, many researchers used symmetric or non-symmetric cryptography techniques in secret messages before embedding them to preserve hiding capacity while providing data protection [22]. The RSA encryption algorithm was used in [23] to encrypt data to ensure security. Another study by [24] integrated steganography and cryptography for safe data transport using the Data Encryption Standard, which used a symmetric key for encryption. However, such cryptographic techniques will affect the secret data size, which increases during encryption. It negatively affects text steganography in terms of capacity and imperceptibility. Therefore, non-sequential or random embedding position techniques are preferred when embedding secret data in the cover text file. An extra layer of a security technique can be added by encrypting secret data in a way that may not increase its size, which then leads to a high-capacity text steganography system. Another issue is increasing the number of secret data bits represented by non-printing or invisible Unicode characters in text documents. As previously mentioned, these characters have limited spaces to embed secret information.

These issues motivate us to develop our proposed method based on the properties of POS tagging with non-printing Unicode Characters and Format-Preserving Encryption. In this paper, the POS tagging works as an indicator for non-printing Unicode Characters (UCs) in Format-based Steganography to improve the hiding capacity and maintain imperceptibility and security properties. The POS properties provide a complete definition for the text content by using word tags to produce a tag list. Words of the eight most common tags will be selected to establish randomized indicator positions in the cover text to hide secret data. In addition, the number of bits in each Unicode character representation is increased to four. A pair of Unicode characters will be used in the proposed method. An extra layer of security technique is applied to encrypt secret data without increasing the size by using Format-preserving encryption. So far, this paper has discussed the text steganography system's capacity, imperceptibility and security issues. One of the goals of the text steganography system is to increase embedding capacity. However, balancing imperceptibility and security is also important with the increased capacity.

The rest of the paper is organized as follows: Section 2 explains related work on text steganography, followed by POS Tagging in Section 3. Section 4 describes Format-Preserving Encryption, and a detailed description of the proposed algorithm is described in Section 5. Section 6 presents a critical analysis that includes obtained results from the experiments and a comparative discussion on former suggested methods. Finally, Section 7 discusses the conclusion and future work of the paper.

2. Related Work

This section examines related works that use the same principle concept as our proposed method, as discussed in Section 5.

A collection of forwarded e-mails is randomly selected as a cover text in a technique proposed in [15], a list of e-mails together with a randomized indexed word dictionary. The cover is an e-mail that gets forwarded. A word dictionary with a randomized index serves as the encryption key for the concealed data in the carbon copy field. A random bitstream (temporary stego key) using the system's time is generated and sent independently using public-key cryptography. The index values of the dictionary terms are randomly generated using this temporary stego key. This approach is safe and secure against typical attacks since it removes noise from the actual e-mail body information. Additionally, it provides an extra degree of protection by randomly generating keys using the system time for the word index values.

The linguistic text steganography method is based on POS tagging proposed by [18]. This method processes secret and text covers with POS tagging to generate a tag list of each secret and cover text. The embedding process replaces each word from the cover text with a word from the secret text based on similarity tags, such as (noun with noun), (verb with verb), etc. Although the cover text remains unchanged in terms of the formatting and the level of similarity is high, it is difficult to control the semantics of the cover text after the embedding process (which is to keep the text's original meaning). At the same time, capacity is limited since the replacement condition is finding the same tag from the secret and cover text.

A multi-keyword coverless text steganography algorithm based on pre-treatment and POS tagging was proposed by [19]. The author used the Chinese Mathematical Expression that represents the Chinese characters. Chinese characters are divided into about 644 basic components. For example, the Chinese character "丛" consists of two parts, "人" and "一". The embedding process starts by dividing the secret message into sequence keyword segments and then randomly changing the sequence. The next step marks the locating keywords using Chinese character Mathematical Expression components. Then, POS is used for embedding the number of keywords inside the cover text and mapping their locations as a key. On the receiver's side, stego-texts are received, and secret messages can be extracted using the mapping locations and random sequence keys.

The author in [25] suggested a text steganography method based on a set of two-letter words from the Oxford Dictionary. This method used a set of two-letter words as markers to secret data embedding. Every two secret text bits are mapped to a certain non-printing Unicode character (UC). Through this procedure, a shared UC mapping table between the sender and recipient is created. The work in [26] suggested a text steganography method based on the Lempel-Ziv-Welch Algorithm and sets of two-letter words that improve the previous method in [25]. The Lempel-Ziv-Welch (LZW) compression algorithm compresses the secret text to minimize its size. A colour-coding approach is used with the LZW compression technique in [27]. The technique used forward mail as a cover medium to obfuscate secret information. The operation first compresses the secret information before disguising it in the e-mail addresses and cover messages. The cover text (or message), which is coloured using a colour-coding table, has the secret data bits included in it. The study's findings showed that the strategy had a higher embedding capability than previous approaches and was less computationally difficult. Additionally, using stego keys greatly improves the security of the suggested method.

A text steganography method in [28] uses e-mail addresses with Huffman compression to improve the capacity of the cover file. The length of an e-mail ID character is used to hide the bits of hidden messages.

Table 1 summarizes the related works of text steganography and their strong and weak points.

Table 1. Related work on text steganography

Refs	**Methodology**	**Strength**	**Weaknesses**
[15]	Secret text embedded using a randomized index and included in a list of e-mail addresses carbon copy (cc) field.	High Capacity High security	Low imperceptibility. Needs a large number of e-mails to embed secret text.
[18]	The embedding process replaces each word from the cover text with a word from the secret text based on POS tag similarity.	High imperceptibility.	Low Capacity. Low security. Need words with the same tags between secret and cover text.

[19]	Used POS to embed the number of secret text keywords generated by Chinese character Mathematical Expression components and mapped their locations as a key.	High imperceptibility. High security.	Low Capacity.
[25]	Each unique Unicode character (UC) is mapped to every pair of secret text bytes and uses a set of two-letter words as indicators for UC to generate a UC mapping table.	Average Capacity.	Low imperceptibility. Low security. Insert a high amount of UC in the cover text that minimizes similarity with fewer bits in map representation.
[26]	compressed the secret text using the LZW algorithm to reduce their size and mapping secret text (two bits) with each UC using a set of two-letter words as indicators for UC to generate a UC mapping table.	High capacity.	Low imperceptibility. Low security.
[27]	compressed the secret text and hid it by using colours in the forward mail based on the colour-coding table.	High capacity. High security.	Low imperceptibility.
[28]	The secret text was compressed with Huffman coding and used e-mail ID characters as cover text to hide the secret text.	High imperceptibility.	Low Capacity. Low security.

3. Part-of-Speech Tagging

Before discussing the proposed model, this section explains the related concepts of POS Tagging. POS tagging is applied to create a list that contains tags of the most common POS categories in English in the cover text file. POS tagging uses morphological analysis to add a POS to each word in a sentence. Morphology is the science of word-forming, or how words are constructed from smaller components. Tokenization, dictionary lookup, and disambiguation are the processes that make up morphological science [18]. A dictionary lookup takes a string and returns a list of lexemes with POS data. In general, there are two types of POS tagging: supervised and unsupervised tagging. A pre-tagged corpus is needed for constructing a POS tagger tool in supervised tagging, while in unsupervised tagging, no pre-tagged corpus is required; instead, advanced computational methods are used to produce a tag set automatically. The POS tagging process starts by breaking down the given text in the natural language processing into the smallest unit in a sentence called a token. Punctuation marks, words, and numbers can be considered tokens [19]. The next step is POS tagging, which labels each word in a sentence with its appropriate POS.

The Penn POS tag set containing 36 Penn sets is shown in **Table 2** [18]. Common POS categories in English are eight Penn tag sets (noun, verb, adjective, adverb, pronoun, preposition, conjunction, and interjection) [29]. These eight POS Penn categories in English are chosen as an indicator among 36 Penn sets of POS for hiding data in the proposed work, which is discussed in the next section.

Table 2. Penn POS Tag Sets

Serial	Tag	Meaning	Serial	Tag	Meaning
01	CC	Coordinating conjunction	19	PRP$	Possessive pronoun
02	CD	Cardinal number	20	RB	Adverb
03	DT	Determiner	21	RBR	Adverb, comparative
04	EX	Existential there	22	RBS	Adverb, superlative
05	FW	Foreign word	23	RP	Particle
06	IN	Preposition	24	SYM	Symbol
07	JJ	Adjective	25	TO	To
08	JJR	Adjective, comparative	26	UH	Interjection
09	JJS	Adjective, superlative	27	VB	Verb, base form
10	LS	List item marker	28	VBD	Verb, past tense
11	MD	Modal	29	VBG	Verb, gerund or present participle
12	NN	Noun, singular or mass	30	VBN	Verb, past participle
13	NNS	Noun, plural	31	VBP	Verb, non-3rd person singular present
14	NNP	Proper noun, singular	32	VBZ	Verb, 3rd person singular present
15	NNPS	Proper noun, plural	33	WDT	Wh-determiner
16	PDT	Pre-determiner	34	WP	Wh-pronoun
17	POE	Possessive ending	35	WP$	Possessive wh-pronoun
18	PRP	Personal pronoun	36	WRB	Wh-adverb

Following is an example with an input text for a POS tagger and the output after using POS tags from **Table 2**.

Input:
“The transmission of confidential data using cover media called steganography.”
Output:
Step 1: text tokenization: “The, transmission, of, confidential, data, using, cover, media, called, steganography”
Step 2: text tagging: “The **DT**, transmission **NN**, of **IN**, confidential **JJ**, data **NNS,** using **VVG**, cover **NN**, media **NNS** , called **VVN**, steganography **NN**”

4. Format-Preserving Encryption (FPE)

The size-increasing problem after encryption processes can be addressed with a unique symmetric encryption technique known as FPE, which is growing acceptance. Compared to well-known encryption methods like AES and DES, this approach is slightly different. [30]. It is a rapidly developing tool for cryptography that ensures data security. Format-preserving encryption seeks to encrypt data without modifying its size or format. Therefore, data will be encrypted differently than traditional encryption, ensuring that the output is the same size and format as the input. There are only two operating modes that the National Institute of Standards and Technology (NIST) recommends: FF1 and FF3. It is suggested to employ the Cipher Block Chaining (CBC) operation that has the ability to encrypt messages of any length using a fundamental block cypher component known as BPS-BC. FF1 and FF3, which stand for BPS-BC (Brier Peyrin Stern) and format-preserving Feistel-based encryption, respectively, were suggested by [30] and [31]. Any format of data blocks can be encrypted using the electronic code book (ECB) operating mode. Even though AES is utilized as the underlying block cypher, operating modes may be considered as a type of FPE block cypher. FPE can also be employed in communication systems when it is necessary to encrypt certain protocols, such as in industrial or military situations or when encrypting specific media [32]. In this paper, in order to improve the security level of stego text files while maintaining the amount of capacity and imperceptibility, a new approach is proposed that focuses on employing format-preserving encryption FF1 mode with POS tagging in a format-based method.

5. Proposed Method

In this section, our proposed method is explained in detail, which consists of three phases: (1) Building a POS Tagging List, (2) Embedding, and (3) Extracting secret messages.

5.1 Building a POS Tagging List

This phase produces outputs: a POS Tagging List (for cover text). The output of this phase is converted into inputs for the embedding phase after checking the embedding capacity of the cover text. Eight Penn tag sets will be used as indicators: nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, and interjections [29]. Fig. 1 shows Algorithm 1, which is to establish the POS list.

```
Algorithm-1:   Building a POS tagging list
Input:         Cover text file (C)
Output:        List of eight most common POS tag tokens

Steps:
    1.  Read C
    2.  For each word in C
        2.1 For each token
                Apply POS Tagger
                Create an entry (location, token, tag)
                Add an entry to a predefined list (most_8list) according to tag (Noun, Verb,
        Adjective, etc.)
        2.2 End For
        2.3 pos_counts = collections.count
    3.  End For
    4.  Most-8 list = pos_counts.most_common (8)
    5.  Return most-8 list
```

Fig. 1. Algorithm to create the list of eight most common POS tag tokens.

5.2 Embedding Phase

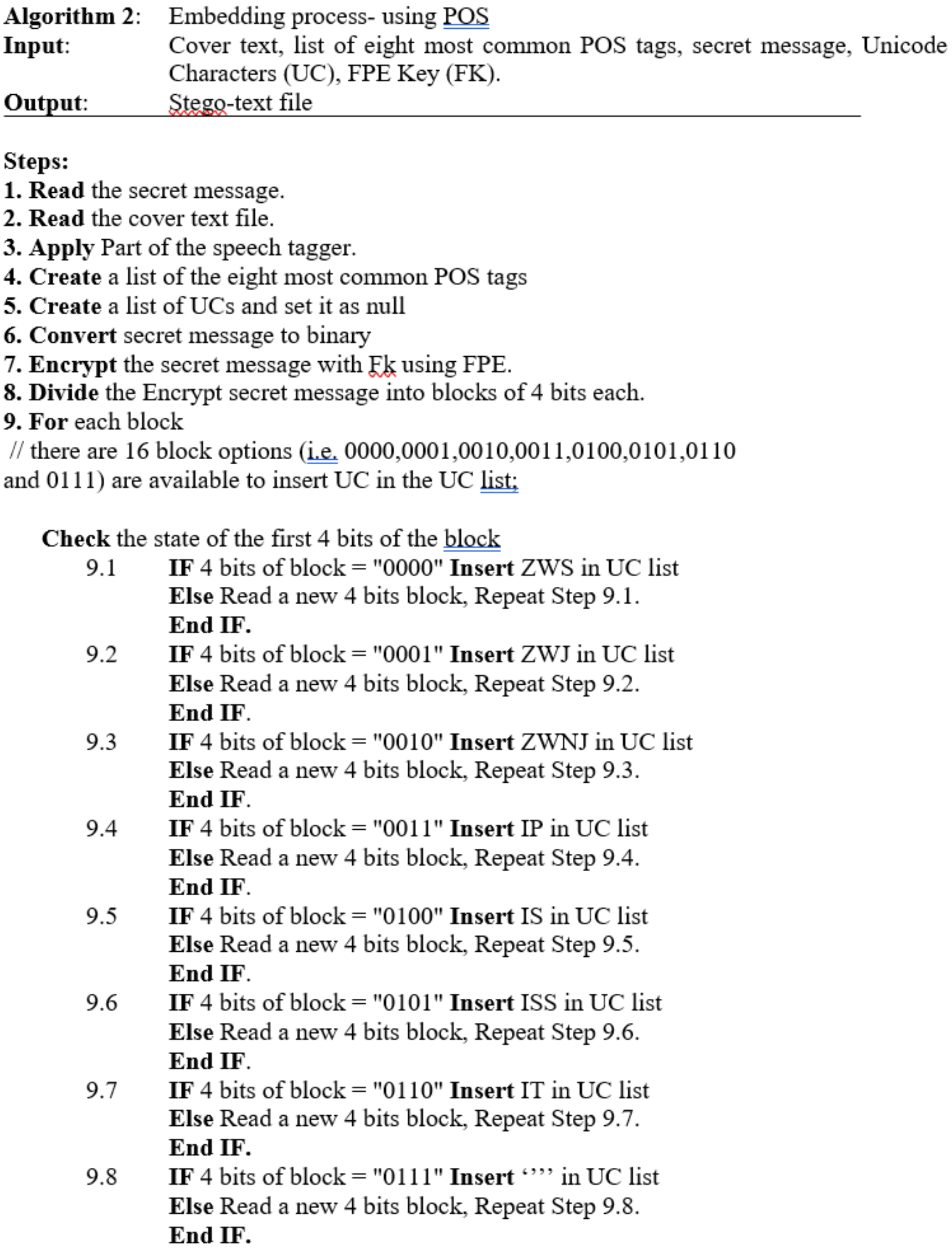
This phase proposes a text steganography technique that exploits the English text redundant data, using POS to resolve the cover text file's limited space for hiding data. This method uses the eight most common POS tags as indicators for hiding secret text data in English scripts using non-printing UCs. Using POS provides randomization in selecting indicators in specific positions in the cover text for UCs representing data bits. Then, FPE is applied to the secret data. FPE is a technique used to encode plaintext in such a way that it can preserve its original length and format. Using FPE can increase the capacity to embed data while simultaneously maintaining the steganography system's security and imperceptibility.

The algorithm in **Fig. 2** (Algorithm 2) uses UCs to hide eight bits of a secret message after each indicator word in the cover text. In Algorithm 2, the cover text is read at the first step and checked for POS tags. Next, create a list of tokens (words) and their tags of the eight most common POS in the text cover. Then, convert the secret message to binary, encrypt the secret message using FPE and divide the binary into blocks of 4 bits. After that, check every 4 bits according to the non-printing UCs mapping defined in **Table 3**. Finally, insert a pair of UCs (two non-printing Unicode characters) after each word (token) of a specific tag according to the eight most common POS tag lists to hide 8 bits in each position. The operation starts with tokens set as noun tags and then moves to the next tokens with another tag until all eight tags tokens in the list are finished. Therefore, the randomization concept will be achieved in specific positions in the cover text since the tokens for each tag are already located and randomly distributed based on the sentence of the cover text. **Fig. 3** shows the Embedding phase flowchart using the proposed method.

Table 3. Non-printing UCs map for hiding 4 bits in each character

Unicode character	**Abbreviation**	**Code**	**Representation**
Zero width space	ZWS	U+200B	0000
Zero width joiner	ZWJ	U+200D	0001
Zero width no- joiner	ZWNJ	U+200C	0010
Invisible plus	IP	U+2064	0011
Invisible separator	IS	U+2063	0100
Inhabit Symmetric Swapping	ISS	U+206A	0101
Invisible Time	IT	U+2062	0110
Empty string	‘’’’	U+2205	0111
Left-To-Right Embedding	LRE	U+202A	1000
Left-To-Right Override	LRO	U+202D	1001
Pop Directional Formatting	PDF	U+202C	1010
Word Joiner	WJ	U+2060	1011
Left-To-Right Isolate	LRI	U+2066	1100
First Strong Isolate	FSI	U+2068	1101

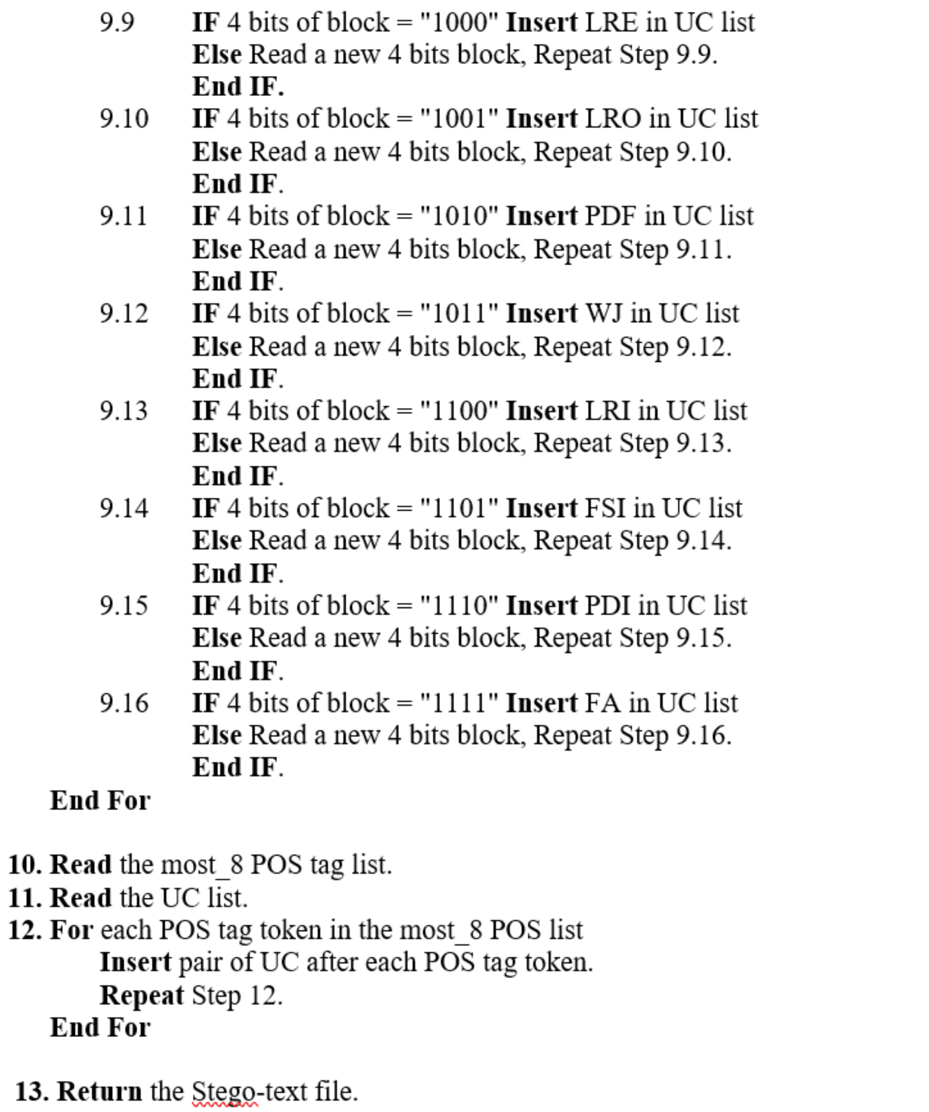
Pop Directional Isolate	PDI	U+2069	1110
Function Application	FA	U+2061	1111



Algorithm 2: Embedding process- using POS
Input: Cover text, list of eight most common POS tags, secret message, Unicode Characters (UC), FPE Key (FK).
Output: Stego-text file

```
Steps:
1. Read the secret message.
2. Read the cover text file.
3. Apply Part of the speech tagger.
4. Create a list of the eight most common POS tags
5. Create a list of UCs and set it as null
6. Convert secret message to binary
7. Encrypt the secret message with Fk using FPE.
8. Divide the Encrypt secret message into blocks of 4 bits each.
9. For each block
 // there are 16 block options (i.e. 0000,0001,0010,0011,0100,0101,0110
and 0111) are available to insert UC in the UC list;

    Check the state of the first 4 bits of the block
        9.1     IF 4 bits of block = "0000" Insert ZWS in UC list
                Else Read a new 4 bits block, Repeat Step 9.1.
                End IF.
        9.2     IF 4 bits of block = "0001" Insert ZWJ in UC list
                Else Read a new 4 bits block, Repeat Step 9.2.
                End IF.
        9.3     IF 4 bits of block = "0010" Insert ZWNJ in UC list
                Else Read a new 4 bits block, Repeat Step 9.3.
                End IF.
        9.4     IF 4 bits of block = "0011" Insert IP in UC list
                Else Read a new 4 bits block, Repeat Step 9.4.
                End IF.
        9.5     IF 4 bits of block = "0100" Insert IS in UC list
                Else Read a new 4 bits block, Repeat Step 9.5.
                End IF.
        9.6     IF 4 bits of block = "0101" Insert ISS in UC list
                Else Read a new 4 bits block, Repeat Step 9.6.
                End IF.
        9.7     IF 4 bits of block = "0110" Insert IT in UC list
                Else Read a new 4 bits block, Repeat Step 9.7.
                End IF.
        9.8     IF 4 bits of block = "0111" Insert ‘’’ in UC list
                Else Read a new 4 bits block, Repeat Step 9.8.
                End IF.
```



```
        9.9     IF 4 bits of block = "1000" Insert LRE in UC list
                Else Read a new 4 bits block, Repeat Step 9.9.
                End IF.
        9.10    IF 4 bits of block = "1001" Insert LRO in UC list
                Else Read a new 4 bits block, Repeat Step 9.10.
                End IF.
        9.11    IF 4 bits of block = "1010" Insert PDF in UC list
                Else Read a new 4 bits block, Repeat Step 9.11.
                End IF.
        9.12    IF 4 bits of block = "1011" Insert WJ in UC list
                Else Read a new 4 bits block, Repeat Step 9.12.
                End IF.
        9.13    IF 4 bits of block = "1100" Insert LRI in UC list
                Else Read a new 4 bits block, Repeat Step 9.13.
                End IF.
        9.14    IF 4 bits of block = "1101" Insert FSI in UC list
                Else Read a new 4 bits block, Repeat Step 9.14.
                End IF.
        9.15    IF 4 bits of block = "1110" Insert PDI in UC list
                Else Read a new 4 bits block, Repeat Step 9.15.
                End IF.
        9.16    IF 4 bits of block = "1111" Insert FA in UC list
                Else Read a new 4 bits block, Repeat Step 9.16.
                End IF.
    End For

10. Read the most_8 POS tag list.
11. Read the UC list.
12. For each POS tag token in the most_8 POS list
        Insert pair of UC after each POS tag token.
        Repeat Step 12.
    End For

13. Return the Stego-text file.
```

Fig. 2. Algorithm for Embedding data using the proposed method.

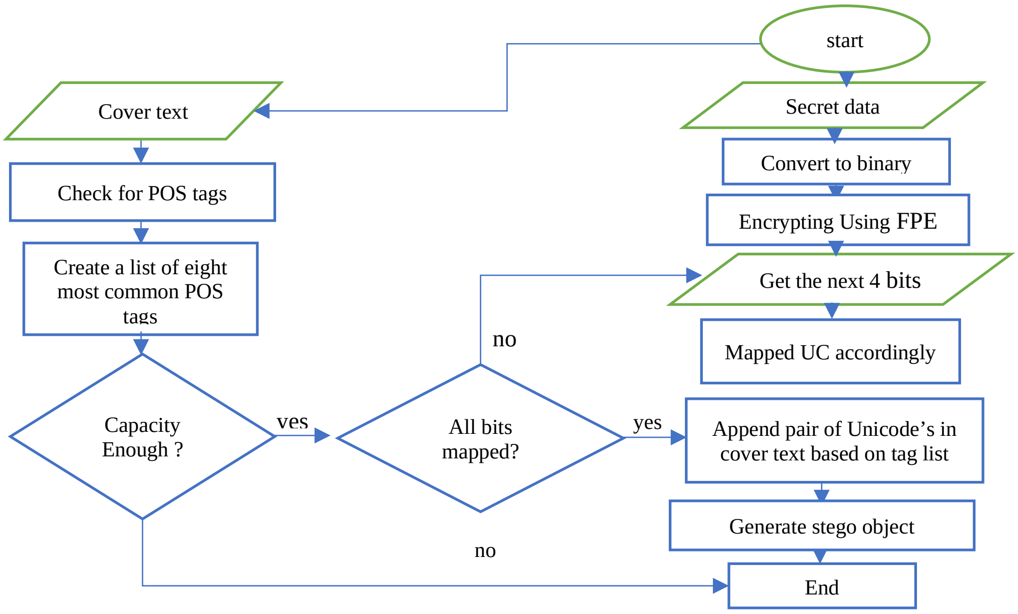
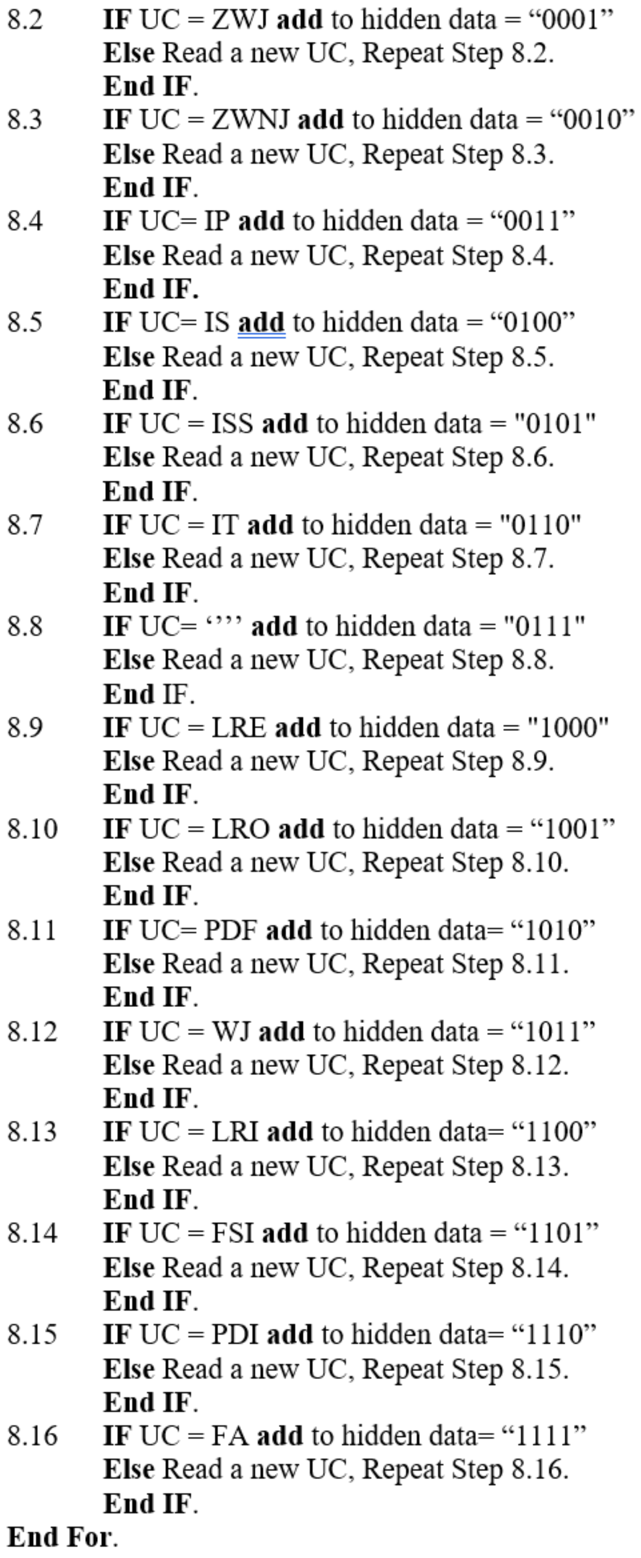


Fig. 3. Flowchart of embedding data using the proposed method

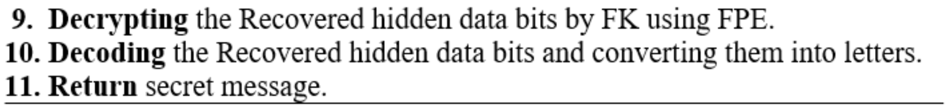
5.3 Extraction phase

POS is also applied to retrieve the secret message encoded from the stego text file. Reading the stego text initiates the extraction process. to check for POS tags and create a list of tokens (words) and tags of the eight most common POS in the text cover. The next step is extracting the alternative Unicode character after each POS tag and mapping each UC according to the UC map defined in **Table 2** to show 4 bits of hidden data. After that, Fk of FPE is used to decrypt the bits of the secret message. Finally, the bits that are recovered are decoded and converted into a letter to reconstruct the secret message text (original). **Fig. 4** shows Algorithm 3, which is the extraction process of the proposed method.

```
Algorithm 3:   Extraction process using POS
Input:         Stego-text file, FPE Key (FK).
Output:        secret message
Steps:
1. Read the cover text file.
2. Set hidden data as null
3. Create a list of UCs
4. Apply Part of the speech tagger.
5. Create a list of the eight most common POS tags
6. Read the POS tags from the list and extract the alternative pair of UCs.
7. Store the UC in the UC list
8. For each UC in the list
   Check the state of the UC
        8.1    IF UC = ZWS  add to hidden data = "0000"
               Else Read a new UC, Repeat Step 8.1.
               End IF.
```



```
8.2     IF UC = ZWJ add to hidden data = “0001”
        Else Read a new UC, Repeat Step 8.2.
        End IF.
8.3     IF UC = ZWNJ add to hidden data = “0010”
        Else Read a new UC, Repeat Step 8.3.
        End IF.
8.4     IF UC= IP add to hidden data = “0011”
        Else Read a new UC, Repeat Step 8.4.
        End IF.
8.5     IF UC= IS add to hidden data = “0100”
        Else Read a new UC, Repeat Step 8.5.
        End IF.
8.6     IF UC = ISS add to hidden data = "0101"
        Else Read a new UC, Repeat Step 8.6.
        End IF.
8.7     IF UC = IT add to hidden data = "0110"
        Else Read a new UC, Repeat Step 8.7.
        End IF.
8.8     IF UC= ‘’’ add to hidden data = "0111"
        Else Read a new UC, Repeat Step 8.8.
        End IF.
8.9     IF UC = LRE add to hidden data = "1000"
        Else Read a new UC, Repeat Step 8.9.
        End IF.
8.10    IF UC = LRO add to hidden data = “1001”
        Else Read a new UC, Repeat Step 8.10.
        End IF.
8.11    IF UC= PDF add to hidden data= “1010”
        Else Read a new UC, Repeat Step 8.11.
        End IF.
8.12    IF UC = WJ add to hidden data = “1011”
        Else Read a new UC, Repeat Step 8.12.
        End IF.
8.13    IF UC = LRI add to hidden data= “1100”
        Else Read a new UC, Repeat Step 8.13.
        End IF.
8.14    IF UC = FSI add to hidden data = “1101”
        Else Read a new UC, Repeat Step 8.14.
        End IF.
8.15    IF UC = PDI add to hidden data= “1110”
        Else Read a new UC, Repeat Step 8.15.
        End IF.
8.16    IF UC = FA add to hidden data= “1111”
        Else Read a new UC, Repeat Step 8.16.
        End IF.
End For.
```



9. **Decrypting** the Recovered hidden data bits by FK using FPE.
10. **Decoding** the Recovered hidden data bits and converting them into letters.
11. **Return** secret message.

Fig. 4. Extraction process of the proposed method.

6. Experimental Analysis and Results

This section evaluates the experimental results of our proposed method. Capacity, imperceptibility, and security are used to evaluate how well the proposed method performs as follows:

6.1 Imperceptibility analysis

In steganography, similarity helps in achieving the imperceptibility criterion. Imperceptibility is one of the most important requirements of steganography systems. It refers to the system's ability to avoid detection. The features of the human visual system (HVS) underpin this necessity. Similarity metrics and the Jaro–Winkler distance can measure a quality stego text's transparency. Similarity metrics analyze the number of identical characters and the transposition of characters in two strings. Jaro-Winkler metric analyses the number of identical characters and the transposition of characters in two strings [18]. The original cover text is compared before and after the embedding procedure to see if the proposed POS method can produce high-quality stego text or if the text's visibility is affected. Equations that calculate the Jaro–Winkler value, which is used to assess the quality of the stego text, are as follows:

$$Jaro\_Winkler(S, C) = Jaro\_Score + \big(L * P * (1 - Jaro\_Score)\big) \qquad (1)$$

$$Jaro\_Score = \frac{1}{3}\left(\frac{m}{Length(s_1)} + \frac{m}{Length(s_2)} + \frac{(m-t)}{m}\right) \qquad (2)$$

Where L is the length of the common prefix at the start of the string up to a maximum of 4, P is the constant scaling factor $(0.1 \geq P \leq 0.25)$, s_1 is the initial string, s_2 is the second string, m is the number of matched characters, and t is the number of transpositions.

As indicated in **Table 4**, the proposed POS method is tested on secret messages embedded in a 10,768-bit cover text file. This setting was used in [25][26].

Table 4. Jaro–Winkler experimental results for the proposed methods

Secret Message	**Size Secret Message (Bit)**	**Jaro-Winkler % (POS)**	**Size Stego File (Bit)**
the import	80	100	11248
the importance and s	160	100	11728
the importance and size of tex	240	100	12208
the importance and size of text data hav	320	99	12688

the importance and size of text data have increase	400	99	13168
the importance and size of text data have increased at an ac	480	99	13648
the importance and size of text data have increased at an accelerating	560	98	14128
the importance and size of text data have increased at an accelerating pace beca	640	98	14608
the importance and size of text data have increased at an accelerating pace because the re	720	98	15088
the importance and size of text data have increased at an accelerating pace because the reliance on	800	98	15568
the importance and size of text data have increased at accelerating pace because the reliance on text based	880	98	16048
Average		**98.8**	

The Jaro–Winkler values of 11 secret messages hidden in the cover file and the stego cover files generated by the proposed method are shown in **Table 4**. The Jaro–Winkler value of 98.8% is achieved by the stego cover text files. However, because each Unicode non-printing character takes three bytes to represent in the text, the size of the stego file will be increased. The performance of the proposed POS method is compared based on the quality of the stego cover to that of techniques used in previous studies [25] and [26]. Both papers used a format-based technique with a sequential embedding pattern by embedding secret bits after each two-letter word, using all available positions in the cover text with UCs. The proposed method gives 95.3% compared to the TWL method in [25] and 98.3% with the LZW method in [26]. The comparison focuses on the stego cover's average Jaro–Winkler value. **Fig. 5** presents comparison results, where the proposed method outperformed the other two.

The proposed method was also compared to the results from [18] in terms of the quality of the stego cover. The technique in [18] also employed POS Tagging in linguistic text steganography. **Fig. 6** presents the comparison result, with the proposed method performing better than the linguistic (POS) method.

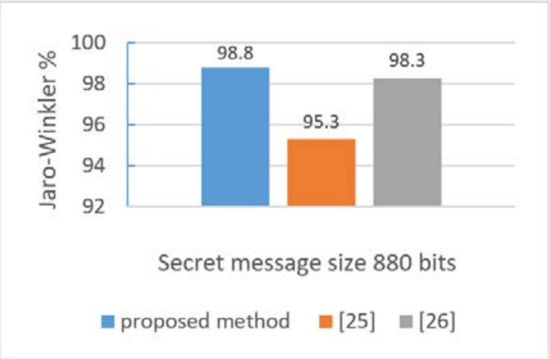


Fig. 5. Comparison of the Jaro–Winkler values of the proposed with [25] and [26] methods

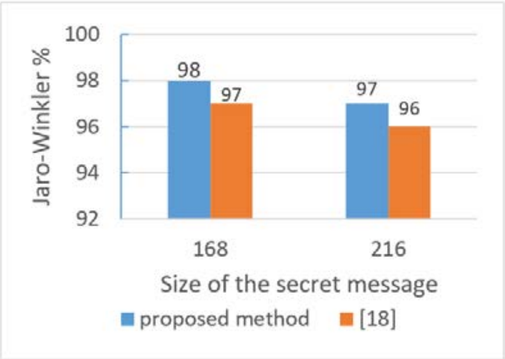


Fig. 6. Comparison of the Jaro–Winkler values obtained by the proposed and [18] method

The comparisons aim to show that the proposed POS method with UCs in the format-based method can achieve a higher result than [25] and [26], which are also based on format-based steganography. Meanwhile, the comparison in [18] was performed in linguistic steganography. Techniques in [25] and [26] insert a higher number of UCs in all available positions in a sequential pattern. In contrast, the linguistic (POS) technique in [18] replaces each word from the secret message with another word from the cover text file, minimizing the similarity between cover and stego text, which is used to count Jaro–Winkler values.

6.2 Capacity analysis

The proposed method is evaluated in terms of capacity ratio and hiding capacity ratio as follows:

6.2.1 Capacity ratio (CR)

Capacity refers to the ability of the cover text to embed secret data. CR can be measured based on the hidden data (bytes) embedded in the cover file. This metric can be mathematically expressed as follows [33], [34]:

$$Capacity\ Ratio\ (CR) = \frac{Size\ \ of\ the\ embedded\ data}{Size\ of\ the\ coverfile} \times 100 \tag{3}$$

An evaluation is performed in terms of its CR using a set of English script resources obtained from http://www.textfiles.com for the first 18 experiments and four experiments from previous studies. In **Table 5**, experiments 19 and 20 are from [26], and Experiments 21 and 22 are from [18].

Capacity significantly depends on the embedding technique. Since every four bits of the secret message are represented in each UC, the POS method embeds eight bits into each POS tag word through a pair of UCs. **Table 5** presents the CR of the proposed POS method.

Table 5. Experimental CR results for the proposed POS method

Experiment	Size of Cover (Byte)	Hidden Data (Byte)	CR (%)
1	2468	312	12.6
2	1812	248	14.6
3	1818	242	13.3
4	1266	173	13.6
5	2131	254	11.9
6	2073	288	13.8
7	1572	194	12.3
8	1886	226	11.9
9	2167	289	13.3
10	1921	243	12.6
11	2887	396	13.7
12	3560	471	13.2
14	4129	570	13.8
15	5341	718	13.4
16	6901	1002	14.5

17	7296	1038	14.2
18	854	122	14.2
19	1346	203	15
20	1208	186	15.3
21	751	128	17.1
22	479	78	16.1

The proposed method obtains a higher CR than the method proposed in the literature [26], which produces a slight variation in the size of invisible characters and hides a maximum of 117 and 96 bytes in the cover files. In comparison, the proposed method can hide (203 bytes and 186) in the same cover files (Experiments #19 and #20 in **Table 5**). **Fig.7** presents the comparison between the two methods.

As explained before, experiments #21 and #22 use POS Tagging in linguistics to hide text [18]. Replacing each word from the secret message with another word that must have the same tag as the cover text will limit the capacity to hide the secret message since the replacement condition uses the same tag from both the secret and cover text. The embeds of [18] are 21 and 27 bytes. Meanwhile, the proposed method embeds 128 and 78 bytes into all space characters, as shown in **Table 5**. **Fig. 8** compares the two cover files used in experiments #21 and #22 and the number of hidden data bits.

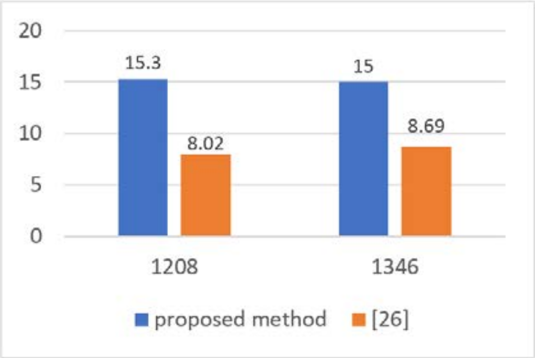


Fig. 7. Comparison in CR between the proposed method and [26]

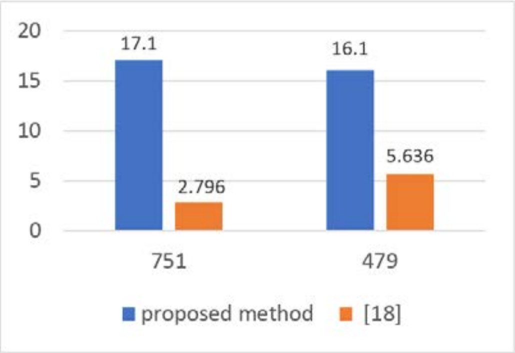


Fig. 8. Comparison in CR between proposed method, and [18]

6.2.2 Hiding capacity ratio (HCR)

A key determining factor in evaluating a text steganography method is hiding capability. The hidden message's ratio size to the stego text's size is known as the hiding capacity HCR. HCR can be mathematically expressed as follows [26],[28]:

$$Hiding\ Capacity\ (HCR) = \frac{Size\ of\ the\ seret\ message}{Size\ of\ the\ stego\ file} * 100 \qquad (4)$$

The proposed method is evaluated in terms of its HCR, as shown in **Table 6**. In the experiment, secret messages of different sizes were embedded in a cover text file.

Table 6. HCR of the proposed method using 13 secret messages

Secret Message	Size Secret Message (Bit)	Stego Size (Bit)	Hiding capacity Ratio (HCR)
the import	80	11248	0.711
the importance and s	160	11728	1.364
the importance and size of tex	240	12208	1.965
the importance and size of text data hav	320	12688	2.522
the importance and size of text data have increase	400	13168	3.038
the importance and size of text data have increased at an ac	480	13648	3.517
the importance and size of text data have increased at an accelerating	560	14128	3.964
the importance and size of text data have increased at an accelerating pace beca	640	14608	4.381
the importance and size of text data have increased at an accelerating pace because the re	720	15088	4.772
the importance and size of text data have increased at an accelerating pace because the reliance on	800	15568	5.139
the importance and size of text data have increased at accelerating pace because the reliance on text based	880	16048	5.484
Average			**3.35**

From **Table 6**, the proposed method has an average HCR value of 3.35%. Increasing the size of the secret message can also increase the HCR value. **Fig. 9** compares the secret message size and HCR using the proposed method.

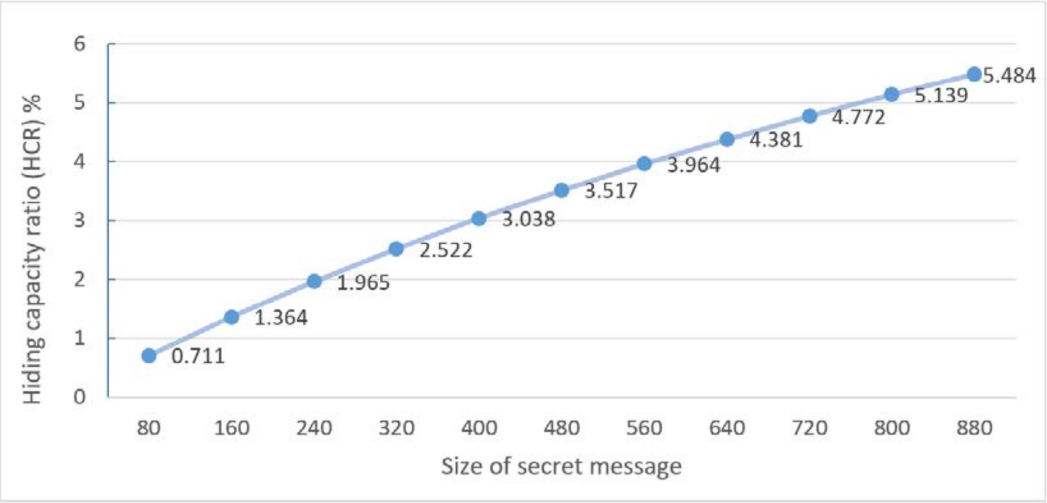
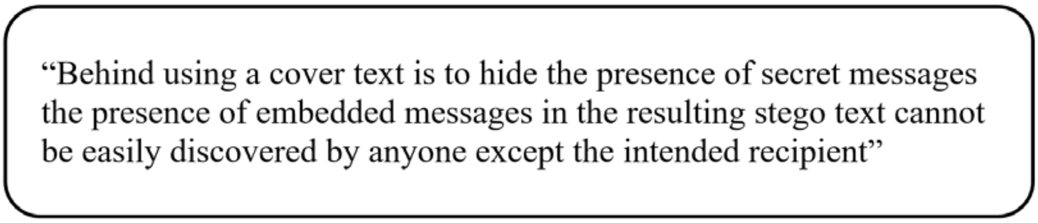


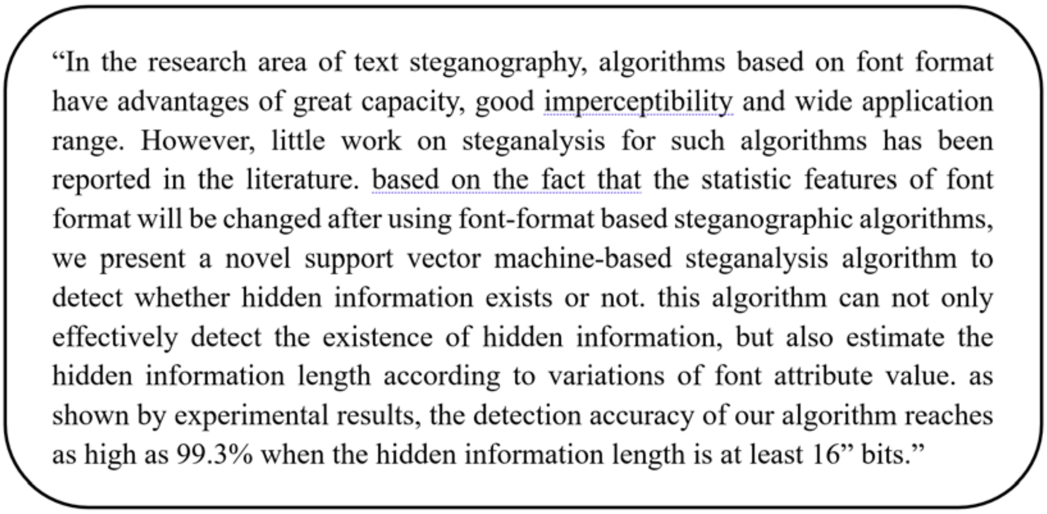
Fig. 9. Comparison between the secret message size and HCR using the proposed method

The proposed method is also compared with several recently proposed approaches, as discussed in the Related Work section. The same resources used in previous studies are applied for the comparison, as shown in **Fig. 10** and **Fig. 11**. The HCR value of the proposed method obtains a capacity of 14.21%, which is superior to the methods proposed by previous studies. **Fig. 12** presents the HCR comparison of the proposed method with previous studies.



"Behind using a cover text is to hide the presence of secret messages the presence of embedded messages in the resulting stego text cannot be easily discovered by anyone except the intended recipient"

Fig. 10. Secret message



"In the research area of text steganography, algorithms based on font format have advantages of great capacity, good imperceptibility and wide application range. However, little work on steganalysis for such algorithms has been reported in the literature. based on the fact that the statistic features of font format will be changed after using font-format based steganographic algorithms, we present a novel support vector machine-based steganalysis algorithm to detect whether hidden information exists or not. this algorithm can not only effectively detect the existence of hidden information, but also estimate the hidden information length according to variations of font attribute value. as shown by experimental results, the detection accuracy of our algorithm reaches as high as 99.3% when the hidden information length is at least 16" bits."

Fig. 11. Cover message

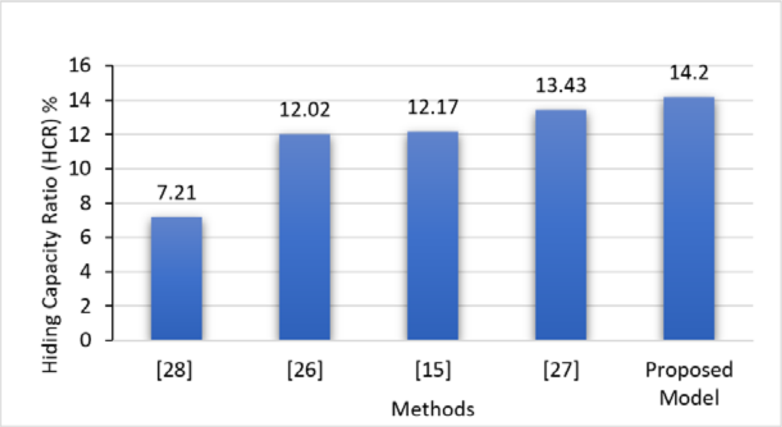


Fig. 12. Comparison (HCR) of the proposed method compared to previous methods

6.3 Security Analysis

Security is one of the important steganography requirements. The security ratio will be small in any text steganography method if excess characters are raised more than the original characters in the cover text [20]. The security ratio is calculated based on the number of increased letters, visible or invisible in the stego-text, compared with the original text. The following equations from [34] are used in the security evaluation. Table 7 presents the security ratio obtained by the proposed method.

$$\text{Discount value} = \frac{\text{Excess character}}{\text{Original character}} * 100 \qquad (5)$$

$$\text{Amount after discount} = \text{Original character} - \text{Discount value} \qquad (6)$$

$$\text{Security ratio} = \frac{\text{Amount after discount}}{\text{Original character}} * 100 \qquad (7)$$

Table 7. Security evaluation of the proposed POS method

Secret Message	**Size Secret Message (Bit)**	**Number of insert characters (UC)**	**Number of original characters in the cover text**	**Security %**
the import	80	20	1313	99.88
the importance and s	160	40	1313	99.76
the importance and size of tex	240	60	1313	99.65
the importance and size of text data hav	320	80	1313	99.53
the importance and size of text data have increase	400	100	1313	99.41
the importance and size of text data have increased at an ac	480	120	1313	99.30
the importance and size of text data have increased at an accelerating	560	140	1313	99.18
the importance and size of text data have increased at an accelerating pace beca	640	160	1313	99.06
the importance and size of text data have increased at an accelerating pace because the re	720	180	1313	98.95
the importance and size of text data have increased at an accelerating pace because the reliance on	800	200	1313	98.83
the importance and size of text data have increased at accelerating pace because the reliance on text based	880	220	1313	98.72
Average				**99.29%**

Table 7 shows the security ratio values of 11 secret messages hidden in the cover file and the generated stego cover files through the proposed method. The stego cover text files achieve a high-security ratio value of 99.29. The average security ratio of the stego file is high since the Unicode characters used in the embedding process represent 4 bits of the secret message. The use of randomized position in the proposed method is based on eight tags only, which means that not all available positions in the cover text will be filled with Unicode characters. Therefore, the number of UCs used will be less even when used in pairs in the proposed method.

7. Conclusion and Future Work

This paper proposes a new format-based text steganography technique using POS tagging and format-preserving encryption. The aim is to use the randomization concept in selecting positions with encryption for data hiding without increasing the secret message size. Therefore, the Unicode characters can be used more efficiently for embedding in cover text. The proposed method uses a 4-bit secret message represented in each Unicode character. A pair of Unicode characters are inserted after each specific tag token in the cover text, providing high embedding capacity and efficiency with minimum effects on imperceptibility and security. Results show that the proposed method has demonstrated improvement in capacity, security, and imperceptibility compared with previous studies. For future research, enhancement can be made by decreasing the secret message size using compression techniques to further increase the capacity of the steganography system.

Acknowledgement

This research was funded by University Kebangsaan Malaysia.

References

[1] M. A. Majeed and R. Sulaiman "An Improved LSB Image Steganography Technique Using Bit-Inverse In 24 Bit Colour Image," *Journal of Theoretical & Applied Information Technology.*, vol. 80, no. 2, Oct. 2015. Article (CrossRef Link)

[2] L. Xiang, R. Wang, Z. Yang, and Y. Liu, "Generative Linguistic Steganography: A Comprehensive Review," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 16, no. 3, pp. 986–1005, Mar. 2022. Article (CrossRef Link)

[3] H. T. Elshoush, M. M. Mahmoud, and A. Altigani, "A new high capacity and secure image realization steganography based on ASCII code matching," *Multimedia Tools and Applications*, vol. 81, no. 4, pp. 5191-5237, Jan. 2022. Article (CrossRef Link)

[4] S. S. Baawi, M. R. Mokhtar, and R. Sulaiman, "A comparative study on the advancement of text steganography techniques in digital media," *ARPN J. Eng. Appl. Sci*, vol. 13, no. 5, pp. 1855-1863, Mar. 2018. Article (CrossRef Link)

[5] A. H Salah, M. Hadwan, A. Aqlan, M. Albazel, F. Alqasemi, and M. Al-Sanabani, “A Survey on Different Arabic Text Steganography Techniques,” in *Proc. of 2021 1st International Conference on Emerging Smart Technologies and Applications (eSmarTA)*, pp. 1-8, Aug 2021. Article (CrossRef Link)

[6] S. Wendzel, L. Caviglione, W. Mazurczyk, A. Mileva, J. Dittmann, C. Krätzer, and T. Neubert, "A revised taxonomy of steganography embedding patterns," in *Proc. of The 16th International Conference on Availability, Reliability and Security*, pp. 1-12, Aug. 2021. Article (CrossRef Link)

[7] B. Yang, W. Peng, Y. Xue, and P. Zhong, "A Generation-based Text Steganography by Maintaining Consistency of Probability Distribution," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 15, no. 11, pp. 4184-4202, Nov. 2021. Article (CrossRef Link)

[8] M. Alkhudaydi, and A. Gutub, "Securing data via cryptography and arabic text steganography," *SN Computer Science*, vol. 2, no. 1, pp. 1-18, Jan. 2021. Article (CrossRef Link)

[9] M. A. Majeed, R. Sulaiman, Z. Shukur, and M. K. Hasan, "A review on text steganography techniques," *Mathematics*, vol. 9, no. 2829, Nov. 2021. Article (CrossRef Link)

[10] M. Shazzad-Ur-Rahman, M. M. H. Ornob, A. Singha, M. S. Kaiser, and N. I. Akhter, "An Effective Text Steganographic Scheme Based on Multilingual Approach for Secure Data Communication," in *Proc. of 2021 Joint 10th International Conference on Informatics, Electronics & Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pp. 1-8, Aug. 2021. Article (CrossRef Link)

[11] Taha, Mustafa Sabah, Mohd Shafry Mohd Rahim, Sameer Abdulsattar Lafta, Mohammed Mahdi Hashim, and Hassanain Mahdi Alzuabidi. "Combination of steganography and cryptography: A short survey," in *Proc. of IOP conference series: materials science and engineering*, vol. 518, no. 5, p. 052003, 2019. Article (CrossRef Link)

[12] S. Al-Nofaie, A. Gutub, and M. Al-Ghamdi, "Enhancing Arabic text steganography for personal usage utilizing pseudo-spaces," *Journal of King Saud University-Computer and Information Sciences*, vol. 33, no. 8, pp. 963-974, Oct. 2021. Article (CrossRef Link)

[13] C. Zhang, C. Lin, P. Benz, K. Chen, W. Zhang, and I. S. Kweon, "A brief survey on deep learning based data hiding, steganography and watermarking," *arXiv Prepr*, Apr 2022. Article (CrossRef Link)

[14] N. Alanazi, E. Khan, and A. Gutub, "Inclusion of Unicode standard seamless characters to expand Arabic text steganography for secure individual uses," *J. King Saud Univ. Inf. Sci*, vol. 34, no. 4, pp. 1343-1356, Apr. 2021. Article (CrossRef Link)

[15] G. Maji and S. Mandal, "A forward e-mail based high capacity text steganography technique using a randomized and indexed word dictionary," *Multimed. Tools Appl*, vol. 79, no. 35, pp. 26549–26569, Jul. 2020. Article (CrossRef Link)

[16] M. A. Majeed, R. Sulaiman, and Z. Shukur, "New Text Steganography Technique based on Multilayer Encoding with Format-Preserving Encryption and Huffman Coding," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 13, no. 12, Dec. 2022. Article (CrossRef Link)

[17] Din, R., Thabit, R. A., Udzir, N. I., & Utama, S, "Traid-bit embedding process on Arabic text steganography method," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 1, pp. 493–500, Feb. 2021. Article (CrossRef Link)

[18] B. Gupta Banik and S. K. Bandyopadhyay, "Novel text steganography using natural language processing and part-of-speech tagging," *IETE J. Res*, vol. 66, no. 3, pp. 384–395, 2020. Article (CrossRef Link)

[19] Y. Liu, J. Wu, and X. Chen, "An Improved Coverless Text Steganography Algorithm Based on Pre-treatment and POS," *KSII Trans. Internet Inf. Syst*, vol. 15, no. 4, pp. 1553–1567, Apr. 2021. Article (CrossRef Link)

[20] A. Ditta, M. Azeem, S. Naseem, K. G. Rana, M. A. Khan, and Z. Iqbal, "A secure and size efficient algorithm to enhance data hiding capacity and security of cover text by using unicode," *J. King Saud Univ. Inf. Sci*, vol. 34, no. 5, pp. 2180-2191, May 2022. Article (CrossRef Link)

[21] Roslan, N. A., Udzir, N. I., Mahmod, R., ZUKARNAIN, Z. A., NINGGAL, M. I. H, & THABIT, R, "Character property method for Arabic text steganography with biometric multifactor authentication using liveness detection," *Journal of Theoretical and Applied Information Technology*, vol. 98, no. 24, Dec. 2020. Article (CrossRef Link)

[22] R. Thabit, N. I. Udzir, S. M. Yasin, A. Asmawi, N. A. Roslan, and R. Din, "A comparative analysis of arabic text steganography," *Applied Sciences*, vol. 11, no. 15, Apr. 2021.

[23] Naqvi, Nuzhat, Aliya Tabassum Abbasi, Rasheed Hussain, M. Aihab Khan, and Basheer Ahmad. "Multilayer partially homomorphic encryption text steganography (MLPHE-TS): a zero steganography approach," *Wireless Personal Communications*, vol. 103, pp. 1563-1585, May. 2018. Article (CrossRef Link)

[24] D. Bhat, V. Krithi, K. N. Manjunath, S. Prabhu, and A. Renuka, "Information hiding through dynamic text steganography and cryptography: computing and informatics," in *Proc. of 2017 international conference on advances in computing, communications and informatics (ICACCI)*, pp. 1826-1831, Sep. 2017. Article (CrossRef Link)

[25] S. S. Baawi, M. R. Mokhtar, and R. Sulaiman, "New text steganography technique based on a set of two-letter words," *J. Theor. Appl. Inf. Technol*, vol. 95, no. 22, pp. 6247- 6255, Nov. 2017. Article (CrossRef Link)

[26] S. S. Baawi, M. R. Mokhtar, and R. Sulaiman, "Enhancement of text steganography technique using Lempel-Ziv-Welch Algorithm and two-letter word technique," in *Proc. of International Conference of Reliable Information and Communication Technology*, pp. 525–537, Sep. 2018. Article (CrossRef Link)

[27] A. Malik, G. Sikka, and H. K. Verma, "A high capacity text steganography scheme based on LZW compression and color coding," *Eng. Sci. Technol. an Int. J*, vol. 20, no. 1, pp. 72-79, Feb. 2017. Article (CrossRef Link)

[28] R. Kumar, A. Malik, S. Singh, and S. Chand, "A high capacity e-mail based text steganography scheme using Huffman compression," in *Proc. of 2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 53-56, Feb. 2016. Article (CrossRef Link)

[29] X. Ding, B. Liu, and P. S. Yu, "A holistic lexicon-based approach to opinion mining," in *Proc. of the 2008 international conference on web search and data mining*, pp. 231-240, Feb. 2008. Article (CrossRef Link)

[30] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma, "A new method for format preserving encryption in high-data rate communications," *IEEE Access*, vol. 8, pp. 21003-21016. Jan. 2020. Article (CrossRef Link)

[31] M. Bellare, P. Rogaway, and T. Spies, "The FFX mode of operation for format-preserving encryption," *NIST Submiss*, vol. 20, no. 19, p. 24, 2010. Article (CrossRef Link)

[32] E. Brier, T. Peyrin, and J. Stern, "BPS: a format-preserving encryption proposal," *Submiss. to NIST*, 2010. Article (CrossRef Link)

[33] R. Thabit, N. I. Udzir, S. M. Yasin, A. Asmawi, and A. Gutub, "CSNTSteg: Color Spacing Normalization Text Steganography Model to Improve Capacity and Invisibility of Hidden Data," *IEEE Access*, vol. 10, Jun. 2022. Article (CrossRef Link)

[34] N. Alanazi, E. Khan, and A. Gutub, "Efficient security and capacity techniques for Arabic text steganography via engaging Unicode standard encoding," *Multimed. Tools Appl*, vol. 80, no. 1, pp. 1403-1431, Sep. 2021. Article (CrossRef Link)

Dr. Mohammed Abdul.M Mohammed.S received his MSc followed by Ph.D from Universiti Kebangsaan Malaysia (UKM). His main area is in Information Security, particularly in Steganography and Applied Cryptography.

Dr. Rossilawati Sulaiman is a Senior Lecturer at the Center of Cyber Security, Universiti Kebangsaan Malaysia (UKM). She did her Bachelor degree at UKM. She received her MSc from the University of Essex and her Ph.D from University of Canberra in 2011. Her research area is in Information Security, particularly in Steganography and Applied Cryptography.

Zarina Shukur is a Professor at the Center of Cyber Security, Universiti Kebangsaan Malaysia. She received a B.Sc degree in Computer from Universiti Kebangsaan Malaysia in 1995. She obtained Ph.D. from the University of Nottingham in 1999. Her current research interests include Blockchain technology, Cyber Security, Formal Method, and Software Engineering.