New Text Steganography Technique based on Multilayer Encoding with Format-Preserving Encryption and Huffman Coding

Mohammed Abdul Majeed[1*], Rossilawati Sulaiman[2], Zarina Shukur[3]
Center for Cyber Security-Faculty of Information Science and Technology,
Universiti Kebangsaan Malaysia (UKM), 43600 UKM, Bangi, Selangor, Malaysia

Abstract—**Steganography is the process of hiding secret data inside other media or cover media. Balancing the requirements for capacity, security, and imperceptibility is the main challenge for any successful steganography system. In text steganography, the data hiding capacity is limited because of the lack of redundant data compared to other digital media, such as images, video, or audio. Other challenges in text steganography are imperceptibility and security. Poor imperceptibility results from the structure of the text file, which is more visually apparent in terms of syntax and grammar than in other media. Low level of security results from the sequential selection of positions for embedding secret data due to insufficient redundant data in a text file. Therefore, an attacker or a third party would notice slight changes in the text file. This paper proposes a new text steganography method that combines cryptography and compression techniques to deal with these issues. This technique is used to conceal secret data to achieve high data hiding capacity in the cover text while maintaining security and imperceptibility. Multilayer encoding and Format-Preserving Encryption (FPE) with Huffman Coding, are applied to secret data before embedding. Invisible Unicode characters are employed to embed secret data into English text files to generate stego files. Results show that the proposed method satisfies capacity and imperceptibility in the cover file by comparing it with previously developed methods.**

Keywords—Text steganography; format-preserving encryption; Huffman coding; unicode characters

I. Introduction

Modern advancements in digital communication are essential to our everyday lives. The utilization of data transfer has substantially expanded because of developments in web-based technologies and the digitization of information. The data transfer includes audio, video, text, and images among individuals and groups, which has become very convenient [1]. However, the allocation of such massive amounts of data over the Internet makes them vulnerable to attacks. Thus, protecting sensitive data has become an important issue requiring immediate solutions. In general, two techniques are used to preserve the security and privacy of sensitive data: cryptography and steganography. One of the most attractive fields for data security is cryptography. This method uses several data encryption techniques to convert sensitive data into ciphertext, which is an incomprehensible format. Another technique for protecting communications during data communication is steganography. Although they have the same goal, steganography and cryptography use different techniques. Steganography, as compared to cryptography, keeps the original data by concealing it in various medium [2] [3].

Both Greek terms "Stegano" and "Graphy," which make up the name "Steganography," have to do with "Cover Writing". The practice of steganography began centuries ago. For instance, Histiaeus employed steganography to transmit messages by inking (tattooing) on the head of his slave, who would travel after the tattoo had grown enough hair to conceal it. Greeks were famous for transmitting secret messages [4] [5]. Since the text has fewer redundant bits than in other cover media like images, music, and video, inserting hidden data in the cover text is the main challenge in text steganography. Due to the scarcity of redundant bits, any little adjustments made to the cover text will be noticeable. Any steganography system must have three main requirements: capacity, security, and imperceptibility [6] [7]. Steganography highly values the imperceptibility of hiding sensitive data in other media. The hiding process is performed without being noticed by human eyes. The concept of "security" refers to "undetectability", where the concealed data is incapable of being found by statistical methods [8]. A steganography system typically seeks to communicate a significant amount of confidential information using the least-covered media to reduce the risk of being discovered when communicating over an unsecured connection [9].

Let's say there is a need to embed a significant amount of sensitive data in a cover file (which will later be called a stego file). In that instance, altering the cover file is more challenging due to the difficulty in achieving imperceptibility and the possibility of distortion. Therefore, the trade-off between hiding capacity with imperceptibility and hiding capacity with security must be identified [10]. The hiding capacity has an inverse relationship with imperceptibility, which means that when large secret data are hidden within a specific size of cover text, inevitably, the stego file will be distorted. This distortion attracts the intruder's attention, and thus the hidden data is noticeable. In general, Unicode characters are used to embed secret data, which requires modification of the cover file. However, because text media suffer from insufficient locations for hiding secret messages, more text will be needed for embedding. In addition, the selection process of the embedding positions of the secret data is performed sequentially. Sequential patterns of the positions make the algorithm vulnerable to attacks [2]. Moreover, text media is

*Corresponding Author.

naturally bounded in terms of syntax and grammar, making text media more visually apparent in the embedding process.

Several researchers have worked on the relationship between hiding capacity and imperceptibility. Compression techniques are used to reduce the hidden secret data size, simultaneously minimizing the amount of modification in the cover file and increasing the imperceptibility [11], such as work in [12], which compressed the secret message using Huffman coding. Secret messages are also compressed in research found in [13] that combined algorithm with minimum-maximum weight and Huffman coding. Meanwhile, in [14], the secret data are concealed in the forward email IDs platform after compressing it with Huffman coding.

Despite implementing the compression algorithms, there are limitations, especially regarding the compression keys. In steganography, the compressed data will be hidden with the compression information, including the decompression keys, which will be shared separately between authorized participants. However, any third party getting the decompression keys could also obtain the secret data. In other cases, compression also led to low imperceptibility, as found in [15].

In addition, research on hiding capacity and security is being done to provide data protection while maintaining hiding capacity. Work in [16] used the RSA algorithm to provide security by encrypting data with minimum modifications in the presence or characters layout. Another research by [17] employs the Data Encryption Standard (DES), a symmetric encryption key, and combines steganography and cryptography for secure data transfer. However, despite these implemented approaches that improved security, cryptography algorithms also increased the secret data size since encryption generally increases the overall size of the data.

Therefore, any new text steganography technique must consider these issues and propose a method that minimizes the secret message size compatible with the available capacity in the cover text. In addition, changes in the cover text should be minimum since embedding secret data in bulk is more noticeable than in other media. More importantly, the embedding technique used to add layers of security must be in such a way that it does not increase the secret message size.

These issues give the motivation to develop the proposed method described further in Section III, which considers the relationship between hiding capacity and imperceptibility on one hand and hiding capacity and security on the other through the combination of cryptography and compression algorithms. This combination has two objectives. Firstly, is to adjust all steganography requirements simultaneously. Secondly is to preserve the trade-off between high hiding capacity and maintaining the imperceptibility of the stego file.

This paper suggests a new text steganography technique using multilayer encoding with FPE and Huffman Coding, which is applied to secret data that will be embedded using invisible Unicode characters inside the cover text. In addition, this paper explains the nature of the text media and the relationship among factors that must be considered when proposing new techniques for text steganography.

The rest of the paper is organized as follows: Section II presents related works. Concepts in Multilayer encoding with format-preserving encryption (fpe) and Huffman coding illustrated in Section III. Proposed method is illustrated in Section IV. Then, Section V presents the experimental results, followed by the conclusion of the work in Section VI. Lastly, recommendations for future research are presented in Section VII.

II. Related Work

This section analyses related works that share the same fundamental concept as the proposed method. Randomization is a concept that can be used as opposed to the sequential concept. This concept can be seen used in text steganography, as found in [18], which used randomization in selecting the forward emails as a cover text. A randomized index-based word dictionary is used to encrypt the carbon copy field that contains hidden data. A temporary stego key from the public key cryptography generates a system time-based random bitstream and is transmitted separately. This method is considered secure against attacks because noises are excluded from the actual email body content. Moreover, randomizing each word of the index values adds an extra layer of security by inserting an 18-bit key generated using the system time in the "date" column of the forward email format. However, the use of public key cryptography increased secret message size and affected negatively on data hiding capacity.

The author in [19] suggested a set of two letters words based on the Oxford dictionary as indicators to hide the secret data represented through non-printing Unicode characters (UC). The proposed approach maps secret data so that every two bits of secret text with a specific UC generates a UC mapping table that will be shared between the sender and receiver. However, the inserting of secret message after each two letters words leads to increase the changes in a cover text and that minimizes a level of imperceptibility. This work is improved in [20] with the Lempel-Ziv-Welch compression algorithm to minimize the secret data size. The secret data are then represented through non-printing UCs to generate the UC mapping table that will be shared between both sender and receiver. However, the capacity is still low due to mapping representation that used a few bits with non-printing UCs in UC mapping table.

Color coding and LZW compression technique are used by [21] that employed the forward mail as a cover text for the secret data embedding process. The secret data is then compressed by the LZW compression technique and embedded inside the cover text by coloring it based in a color-coding table. However, the capacity is still low due to limitation in mapping representation in color-coding table. Another work that used the Huffman compression can be found in [22] to minimize the secret data. A specific number of characters in an email ID indicates the bits of hidden messages. However, the capacity is still low due to mapping representation that used one bit only in color-coding table.

III. Concepts in Multilayer Encoding with Format-Preserving Encryption (FPE) and Huffman Coding

The rapid increase in the number of covert activities in communication networks has intensified the need to devise an efficient data-hiding method to protect secret information from malicious attacks. One possible solution to this problem is to combine steganography techniques with encryption and compression techniques. As previously mentioned in Section II, several encryptions and compression techniques have been proposed, each with unique advantages and disadvantages. An encryption and compression technique that can provide a high security and compression ratio while maintaining an acceptable imperceptibility for the output file must be adopted. To this end, the proposed model applies multilayer encoding with (FPE) and Huffman Coding, which seeks to encode by encrypting and compressing the secret message before embedding.

This model consists of encryption and compression techniques (as shown in Fig. 1), which are applied to encrypt and compress the secret message and consequently increase the security ratio and hiding capacity.

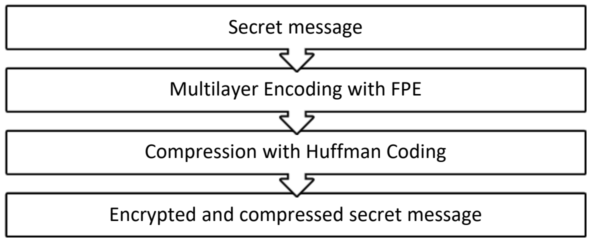
Secret message
Multilayer Encoding with FPE
Compression with Huffman Coding
Encrypted and compressed secret message

Fig. 1. Multilayer encoding with FPE and Huffman coding process

A. Format-Preserving Encryption (FPE)

A novel symmetric encryption method known as FPE is gaining popularity as a solution to previously mentioned problems. This method differs slightly from traditional encryption protocols like AES and DES [23] [24]. It is a rapidly growing cryptography tool that serves the purpose of secrecy in cryptography by ensuring data security. As the name suggests, format-preserving encryption aims to encrypt data without changing its size or format. It involves encrypting data so that the output matches the input's size and format, which offers several advantages over traditional encryption. Feistel structure-based schemes called FF1 and FF3 are used to implement FPE algorithms. The National Institute of Standards and Technology (NIST) advises only two operating modes: FF1 and FF3. A basic block cipher component called BPS-BC is proposed to be used in Cipher Block Chaining (CBC) mode to encrypt messages of any length. FF1, known initially as format-preserving Feistel-based encryption (FFX), was proposed by [25], and FF3, corresponding to the BPS-BC (Brier Peyrin Stern), was proposed by [26]. Fig. 2's non-binary Feistel structure is the foundation for both operating modes. Both FF1 and FF3 can use electronic code book (ECB) mode to encrypt data blocks.

Despite using AES as the underlying block cipher, operating modes may be considered a type of FPE block ciphers. The encryption of data with changeable forms, such as Primary Account Numbers (PANs) or Social Security Numbers (SSNs) that are not in binary format, is an example of a practical use of FPE [26][27]. FPE can also be used in communication systems when it's crucial to encrypt specific protocols, such as in industry or the military or when encrypting particular image formats [28].

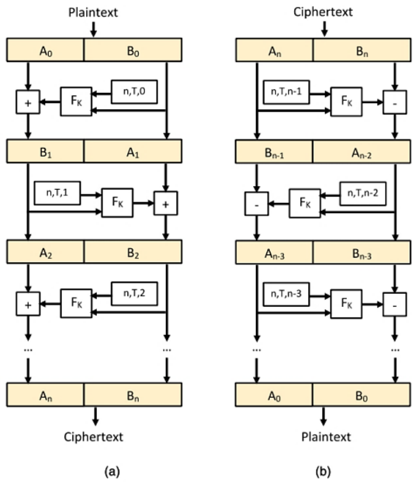

Fig. 2. Generic cipher structure for format preserving encryption: (a) ciphering, (b) deciphering

The present study focuses on the concept of using format-preserving encryption FF1 mode. Data encryption must be considered when developing a conceptual framework because the technique of data encryption leads to an increase in the level of security with maintaining the capacity ratio of stego text files.

B. Compression using Huffman Coding

This section examines the data compression techniques that benefit from current compression techniques related to text files. These techniques can also be combined with steganography to improve the proposed solution. Data compression refers to the process of encoding input data by using a few bits representing the input's original size [29]. In other words, data compression denotes information in a compact format. The data's structure must first be determined to construct such small representations. These data may refer to characters in a text file generated by other processes. Compressed data are only used when both communicating parties are informed of the coding scheme, similar to any other type of communication. Compression is important since it lowers the use of costly resources.

Data compression techniques are highly recommended to increase the hiding capacity in cover files. These techniques, which are popular in computing, require the data transmitted over the Internet to be as compact as possible. Named after the late David Huffman in the 1950s, Huffman coding is a data compression technique that utilizes the greedy algorithm and achieves remarkable savings in capacity (ranging from 20% to 90%).

A variable-length code is employed in Huffman compression coding. Utilizing lower-frequency characters with more extended codes is also preferable in this case. The encoding process involves assigning a numerical string based on the frequency of characters. The Huffman code algorithm takes a string of symbols and transforms them into a varying-length binary string. Then, a binary tree is generated for decoding the binary strings. Each binary character is assigned with 1 for the right child and 0 for the remaining child routes.

IV. Proposed Method

This section describes the proposed method. There are four main components: the encoding phase, embedding phase, extracting phase, and decoding phase, as shown in Fig. 3.

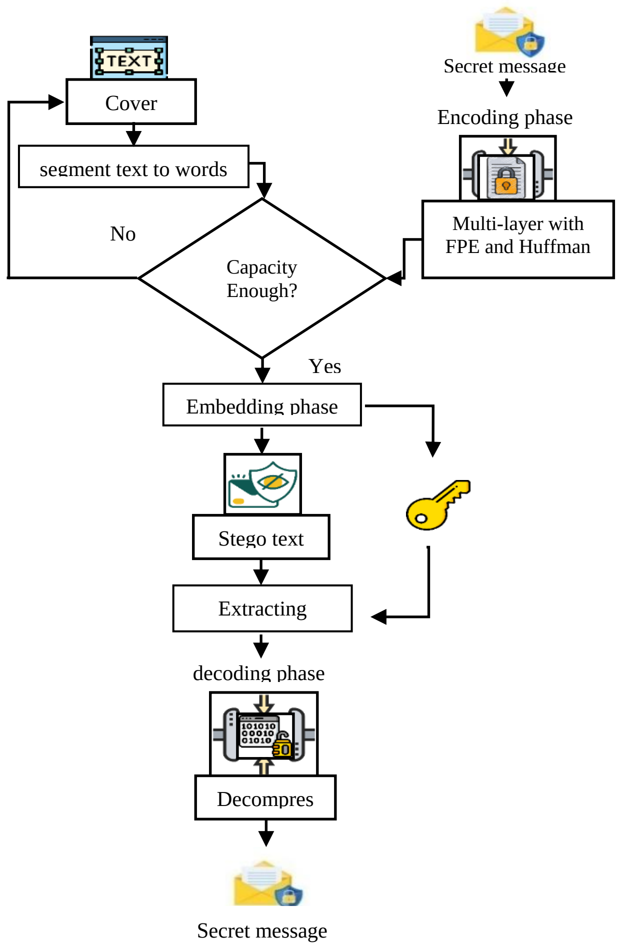

Fig. 3. Block diagram of the proposed scheme

A. Encoding Phase

In this phase, a combination of compression and security is achieved by implementing the multilayer encoding with FPE and Huffman Coding for compression, where secret messages are encoded using FPE. This stage is called multilayer encoding because it involves a few layers of encoding, such as from text to binary, then binary to decimal, and finally applying mod 26 to obtain the final encoding. FPE and the multiple encodings provide multilayer security to the secret message before its actual embedding. Multilayer encoding also reduces the size of hidden messages while providing more hiding spaces, as shown in Algorithm 1 of the multilayer encoding with FPE (Fig. 4). Next, Fig. 5 depicts the algorithm for data compression using Huffman Coding. This algorithm starts by computing the frequency of occurrence of each data (output code from FPE), which is calculated in the input stream. These codes are then arranged from the highest frequency to the lowest.

```
Algorithm 1: multilayer encoding with FPE
    Input: Secret message ($M_S$), FPE Key (Kp )
    Output: $EB_i$ , $k_S$
Steps:
  1. Select Kp
  2. For each Char in secret message ($M_S$)
         Convert $M_S$ into Binary.
     End For
  3. Convert the binary value of $M_S$ into decimal ($M_d$)
  4. Divide $M_d$ with x , $Bi = \frac{M_d}{x}$ // x is random length of the
     divisor.
  5. Encrypt $B_i$ with Kp using FPE.
         $EB_i = Eph(B_i,(Kp) \bmod n$
  6. Take mod 26 of the Encrypted Block
         ($B_i$) $EB_i = B_i$ mod26.
7. Store the reminder bits of mode 26 as the second key, $k_S$ for
each block.
```

Fig. 4. Algorithm 1: multi-layer encoding with FPE

```
Algorithm 2: Huffman message compression
Input: Output codes from the multilayer encoding with FPE
Output: Compressed secret message (CM)

Steps:
1. Read the output codes
2. Compute the frequency of occurrence of each output code in a list
   of (output code, frequency).
3. Arrange the list from the highest to the lowest frequency.
4. Calculate summation of the last two frequency numbers.
5. Rearrange the values in descending order based on their frequency
   numbers.
6. If there is more than one element in the list
       Repeat Step 3.
       End-If
7. Construct a Huffman tree by assigning the value (0,1) to each pair
   of branches in the tree.
8. Construct the final table (Huffman coding) that contains the leaf
   nodes (output codes) and their respective codes according to the
   Huffman tree.
9. Generate the compressed secret message (CM) by rewriting the
   output codes using the table in Step 7.
```

Fig. 5. Algorithm 2: Compression with Huffman coding

In Algorithm 1 (Fig. 4), in line 1, the algorithm selects the value of the FPE key (Kp). The block encoding and encryption processes are shown in lines 2–8. Firstly, each character of the secret message (Ms) is converted into binary. Next, in line 4, a binary string is converted into a decimal string Md, and Md is divided into equal blocks with a random length of the divisor (x), shown in line 5. After that, the encryption phase (labelled as *Eph*) is applied on each block (Bi) with Key (Kp), $EB_i = Eph(B_i, k_p) \bmod 26$ (line 6). In line 7, mod 26 on EB_i is calculated, and finally, the k_S values are determined by collecting the reminder bits of mode 26 for each block (line 8).

The k_S values are used to regenerate encrypted blocks in the extraction process.

In Algorithm 2 (Fig. 5), Huffman coding begins with the input, representing the output codes of the multilayer encoding with the FPE algorithm that need to be compressed. The algorithm then computes the frequency of occurrence of each output code in the input stream. These codes are subsequently arranged from the highest to the lowest frequency. The two codes with the lowest frequencies are treated as children of the node, and the parent node comprises the total frequency of these two child nodes. This node is then inserted back into the list, and the list is rearranged.

The process of applying Multilayer Encoding with FPE and Huffman coding is shown in the following example:

Suppose the Secret message is: "**Universiti Kebangsaan Malaysia**".

1) Multilayer encoding with FPE:

Step 1: Select the encryption key "Kp" value for FPE. Assume Kp=1200.

Step 2: Convert the secret message into random binary stream blocks. The obtained binary blocks are given below:

01010101 01101110 01101001 01110110 01100101 01110010 01110011 01101001 01110100 01101001 00100000 01001011 01100101 01100010 01100001 01101110 01100111 01110011 01100001 01100001 01101110 00100000 01001101 01100001 01101100 01100001 01111001 01110011 01101001 01100001

Step 3: Convert each binary block into a decimal stream to obtain the following decimal stream:

8511010511810111411510511610532751019897110103115979711032779710897121115105 97

Step 4: Convert the decimal stream into random blocks:

8511010511 8101114115 1051161053 2751019897 1101031159 7971103277 9710897121 11510597

Step 5: Encrypt each decimal block using FPE with Kp=1200; the resultant encrypted blocks are given like the following:

1909100954 6927975788 1398002705 6714036020 1293204947 2415200591 2308156747 39114904

Step 6: Apply mod 26 on each encrypted block, and the remaining bits of each block after applying the mod 26 equation are stored as the second key, Ks.

The results of applying mod 26 on each encrypted block are as follows:

[**20 6 21 16 21 11 13 10**]

2) Applying huffman coding:

Step1: Read the output codes of multilayer encoding with FPE, which are:

[**20, 6, 21, 16, 21, 11, 13, 10**].

Step 2: Construct the final table (Huffman coding table) that contains the leaf nodes (output codes), as shown in Table I.

TABLE I. HUFFMAN CODING TABLE

Node	20	6	21	16	21	11	13	10
Codeword	110	1000	11	101	00	010	011	1001

Step 3: Generate the compressed secret message by rewriting the output codes using the table of Huffman coding.

Original secret message: (20, 6, 21, 16, 21, 11, 13, 10)

Compressed secret message: (110, 1000, 11, 101, 00, 010, 011, 1001).

The length of the compressed secret message, as computed using the Huffman algorithm, is 24 bits.

B. Embedding Phase

By using the invisible Unicode characters (UC), the embedding process aims to resolve the cover text limitation in data hiding capacity. In the proposed method, the embedding process is performed using eight Unicode characters to embed the secret data. These characters are inserted into spaces between words of the cover text. This phase proposes a text steganography technique using the property of data redundancy in the English text to improve the imperceptibility of hidden information. This method uses eight invisible Unicode characters UC by mapping three bits in each UC, as shown in Table II.

TABLE II. UC MAP FOR HIDING 3 BITS IN EACH CHARACTER

Unicode characters	**Abbreviation**	**Representation**
Zero width character	ZWC	000
Zero width joiner	ZWJ	001
Zero width no- joiner	ZWNJ	010
Invisible plus	IP	011
Invisible separator	IS	100
Inhabit Symmetric Swapping	ISS	101
Empty string	‘’’	110
Left-To-Right Embedding	LRE	111

The algorithm uses UC to hide three secret message bits after each word in the cover text. The first step of Algorithm 3 is to read the cover text and start segmenting text into words. Then, secret messages are converted to binary, which is divided into blocks of three bits. Next, every three bits are checked according to UC mapping, as defined in Table II. Then the alternative Unicode character is inserted after each word of cover text. Fig. 6 shows the Embedding algorithm.

```
Algorithm 3: Embedding process
Input: cover text, secret message, UC
Output: Stego-text file

Steps:
1.  Read the secret message.
2.  Read the cover text file.
3.  Segment cover text to words
4.  Divide the hidden data into blocks of 3 bits each.
5.  For each block in a secret message
        // there are 8 block options (i.e
        000,001,010,100,101,110     and 111) are
        available to insert UC after each word.
        Check the state of the first 3 bits of block
    5.1  IF 3 bits of block = "000" Insert ZWC after it
         Else Read a new word, Repeat Step 5.1.
    5.2  IF 3 bits of block = "001" Insert ZWJ after it
         Else Read a new word, Repeat Step 5.2.
         End IF.
    5.3  IF 3 bits of block = "010" Insert ZWNJ after it
         Else Read a new word, Repeat Step 5.3.
         End IF.
    5.4  IF 3 bits of block = "011" Insert IP after it
         Else Read a new word, Repeat Step 5.4.
         End IF.
    5.5  IF 3 bits of block = "100" Insert IS after it
         Else Read a new word, Repeat Step 5.5.
         End IF.
    5.6  IF 3 bits of block = "101" Insert ISS after it
         Else Read a new word, Repeat Step 5.6.
         End IF.
    5.7  IF 3 bits of block = "110" Insert ''' after it
         Else Read a new word, Repeat Step 5.7.
         End IF.
    5.8  IF 3 bits of block = "111" Insert LRE after it
         Else Read a new word, Repeat Step 5.8.
         End IF.
    End for
6.  Return Stego-text file
```

Fig. 6. Algorithm 3: Data embedding

C. Extraction Phase

The extraction phase is the third phase implemented in the proposed method. The stego text file is used as input. The extraction phase aims to extract the hidden data from the stego text file by retrieving each invisible UC. The extraction starts by reading the stego text. Next, the alternative Unicode character after each word is extracted. Then, each UC is mapped as defined in Table II to show the three bits of hidden data. Finally, return the recovered encoded secret message bits for the Decoding phase to reconstruct the secret message. Fig. 7 summarizes the extraction process.

```
Algorithm 4: Extraction process
Input: Stego-text file
Output: Recovered encoded Secret message

Steps:
    1.  Read the stego text file.
    2.  Set secret data is null
    3.  Segment stego text to words.
    4.  Read each word and extract the alternative UC.
    5.  For each UC
        5.1  IF UC = ZWC add to hidden data = "000"
             Else Read a new UC, Repeat Step 5.1
             End IF.
        5.2  IF UC = ZWJ add to hidden data = "001"
             Else Read a new UC, Repeat Step 5.2.
             End IF.
        5.3  IF UC = ZWNJ add to hidden data ="010"
             Else Read a new UC, Repeat Step 5.3.
             End IF.
        5.4  IF UC= IP add to hidden data = "011"
             Else Read a new UC, Repeat Step 5.4.
             End IF..
        5.5  IF UC= IS add to hidden data ="100"
             Else Read a new UC, Repeat Step 5.5.
             End IF.
        5.6  IF UC = ISS add to hidden data ="101"
             Else Read a new UC, Repeat Step 5.6.
             End IF.
        5.7  IF UC = '''  add to hidden data ="110"
             Else Read a new UC, Repeat Step 5.7.
             End IF.
        5.8  IF USC= LRE  add to hidden data ="111"
             Else Read a new UC, Repeat Step 5.8.
             End IF.
      End for
  6.  Return secret message.
```

Fig. 7. Algorithm 4: Data extraction

D. Decoding Phase

The decoding phase includes two processes, namely: the decompression and deciphering of the secret text. The encoded secret message retrieved in the extraction phase is used as the input, while the retrieved secret message text is produced as the output.

1) Decoding of Multilayer Encoding with Format-Preserving Encryption and Huffman code: An encoded secret message retrieved from the extracting phase is decoded using the Huffman coding table to return the indexes. The Huffman coding table is transmitted as a key file. Decoding the retrieved encoded secret message and returning the indexes require rebuilding the Huffman tree based on the Huffman coding table. This process iterates through the binary encoded data.

The process starts traversing from the root until a leaf is found to find the characters that correspond to the current bits. The node on the left of the tree is then approached if the current bit is 0. If the bit is 1, the approach is made to the right node of the tree. When traversing, a leaf node is reached, and its character is displayed. After that, the encoded data is iterated till the end. The significant advantage of Huffman coding is that, although each character is coded with various bits, the receiver will automatically locate the character in order.

In FPE, the encoded data stream is decoded following the same principles applied in the encoding process. The process starts by obtaining the secret message bits from the decompression process with the Huffman code and then applying the decryption process with Ks, which is the key established after applying mode 26 in the multilayer process in the encoding stage. The next step is decrypting the secret message bits from FPE using the Kp key, converting secret bits from decimal to binary to rebuild the characters using ASCII code at the end of this algorithm (Fig. 8).

Algorithm 5: Decoding Multilayer Encoding with Format Preserving Encryption
Input: Encoding data stream , k_S , k_p
Output: Recovered Secret message

```
Steps:
1.  Read Encoding data stream.
2.  For each number in stream i.e. (20,6,21,16, 21,11,13,10).
            get ND using $k_S$.
            get NS using , $k_p$.
    End for
3.  For each number in ND stream
            add number in Temp T
            IF $T \geq 64$ then
            Add to DI list (T)
            T=0
            Else
            IF T =32 then
            Add to DI (T)
            T=0
            Endif
    End for
4.  For each number in DI list
            convert to binary
            retrieve secret message SM
    End for
5.  Return secret message.
```

Fig. 8. Algorithm 5: Data decoding

For example:

The retrieved message: (**20,6,21,16, 21,11,13,10**)

Step 1: decrypt (**20,6,21,16, 21,11,13,10**) using $\boldsymbol{k_S}$ which is mod 26 *reminder bits* of each number in encode stage.

The results of decryption will be:

1909100954	6927975788	1398002705
6714036020	1293204947	2415200591
2308156747	39114904	

Step 2: decrypt each block using FPE with Kp; the resultant decrypted blocks will be:

8511010511	8101114115	1051161053
2751019897	1101031159	7971103277
9710897121	11510597	

Step 3: combine blocks of numbers to accomplish the stream of decimal numbers. The stream will be separated according to original decimal numbers that represented each binary of secret bits:

85110105118101114115105116105327510198971101031159797110327797108971211151059 7.

Step 4: steps to accomplish the separation according to the original of the decimal numbers:

- take two numbers.
- **If** the summation of the two numbers equals 32, then separate
- **else-if** the sum of the two numbers equals or is more than 64, then separate
- **Else** take three numbers and separate

where 32 represents the word space in decimal, which are separate words. This condition identifies the word spaces in the algorithm. At the same time, since each letter in the English language starts with 64 when represented in decimal, and the numbers reflect the secret message letters in decimal, the retrieval numbers must be 64 or more. This condition could be satisfied with two or three numbers from the string while separating the secret message. The output from this stage will be (85 110 105 118 101 114 115 105 116 105 32 75 101 98 97 110 103 115 97 97 110 32 77 97 108 97 121 115 105 97).

Step 5: Convert each decimal number into binary and then retrieve the secret message from the binary. The binary representation will be as follows:

01010101	01101110	01101001	01110110	01100101
01110010	01110011	01101001	01110100	01101001
00100000	01001011	01100101	01100010	01100001
01101110	01100111	01110011	01100001	01100001
01101110	00100000	01001101	01100001	01101100
01100001	01111001	01110011	01101001	01100001

The secret message will be "**Universiti Kebangsaan Malaysia**".

V. Experimental and Analysis

In this section, we analyze the experimental results of our proposed method. The performance of the proposed method is measured in terms of capacity and imperceptibility. The hiding capacity is a major crucial parameter for analysis of the text steganography algorithm performance. We used the Jaro-Winkler distance, as used in [20], which is a similarity metric for the cover text and stego cover (1) and (2). The similarity metrics were used to calculate how similar two strings were to one another, with (0) indicating a difference and (1) indicating equal matching or imperceptibility of strings.

$$Jaro\_Winkler(S,C) = Jaro\_Score + \left(L * P * (1 - Jaro\_Score)\right) \quad (1)$$

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

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 $(0.1 \geq P \leq 0.25)$, m is the number of matched characters, s_1 is the first string, s_2 is the second string, and t is the number of transpositions. As in [18, 21], hiding capacity is defined as the hidden data size relative to the size of the stego cover. Formulation (3) can be used for calculating the hiding capacity.

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

The proposed method is applied to the cover file as shown in Fig. 9 and secret messages shown in Fig 10, which are divided into twelve samples to be embedded into the cover file. Table III shows the results of the experiments, which include the secret messages, size secret message (n) and Jaro-Winkler (JW), for the proposed method and the comparison with previous related studies.

"The loss of tree cover as a result of forests being cleared for other land uses such as farming and logging is called deforestation. Deforestation activities affect carbon fluxes in the soil, vegetation, and atmosphere. However, logging can also lead to carbon emissions if the surrounding trees and vegetation are damaged. Deforestation is defined as the destruction of forested land. It is a major problem all over the world. The causes of deforestation vary from place to place. The most common causes, however, are logging, agricultural expansion, wars, and mining. Deforestation has been the cause of many problems facing the world today such as erosions, loss of bio diversity through extinction of plant and animal species, and increased atmospheric carbon dioxide. In this paper, I will present that we can reduce deforestation by moving from physical letter mail to electronic mail. From the ancient era physical letter mail has come, till now it is going on, but, on the other side due to this everyday trees are being cut i.e., deforestation is taking place by the paper industry, hence increasing CO2 emission and global warming. In place of physical letter mail, we can use electronic mail which will definitely do some reduction in deforestation. There are critical effects of deforestation."

Fig. 9. Cover text (1)

"The importance and size of text data have increased at accelerating pace because the reliance on text based web01234."

Fig. 10. Secret message (1)

As shown in Table III, the experimental results indicate that the proposed method can be applied to embed the secret message in the cover file. The Jaro-Winkler similarity score is 0.984 which is higher than in the two previous studies.

TABLE III. JW OF THE PROPOSED METHOD COMPARED WITH RELATED STUDIES

Secret Message	**Message size (Bit)**	**JW [15]**	**JW [16]**	**JW Proposed work**
the import	80	0.99	1	1
the importance and s	160	0.98	1	1
the importance and size of tex	240	0.98	0.99	0.99
the importance and size of text data hav	320	0.97	0.99	0.99
the importance and size of text data have increase	400	0.96	0.99	0.99
the importance and size of text data have increased at an ac	480	0.95	0.98	0.99
the importance and size of text data have increased at an accelerating	560	0.95	0.98	0.98
the importance and size of text data have increased at an accelerating pace beca	640	0.94	0.98	0.98
the importance and size of text data have increased at an accelerating pace because the re	720	0.94	0.98	0.98
the importance and size of text data have increased at an accelerating pace because the reliance on	800	0.93	0.97	0.97
the importance and size of text data have increased at an accelerating pace because the reliance on text based	880	0.93	0.97	0.97
the importance and size of text data have increased at an accelerating pace because the reliance on text based web 01234.	936	0.92	0.97	0.97
Average		0.953	0.983	0.984

Next, the performance in capacity is compared with related studies that use the same secret message, as shown in Fig. 11 and the cover file in Fig. 12. Results show that the proposed method also scores higher in hiding capacity results. Table IV compares the hiding capacity of the proposed method with existing techniques. The proposed method achieved 18.4% capacity, which performed better than the other techniques for the same cover text and the secret message. Fig. 13 presents the bar chart comparison of the capacity that is listed in Table IV.

"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. 11. Secret message (2)

“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. 12. Cover text (2)

TABLE IV. CAPACITY OF PROPOSED METHOD COMPARED WITH RELATED STUDIES

Method	**Hiding Capacity Ratio %**
[22]	7.21
[20]	12.02
[18]	12.17
[21]	13.43
Proposed Method	18.4

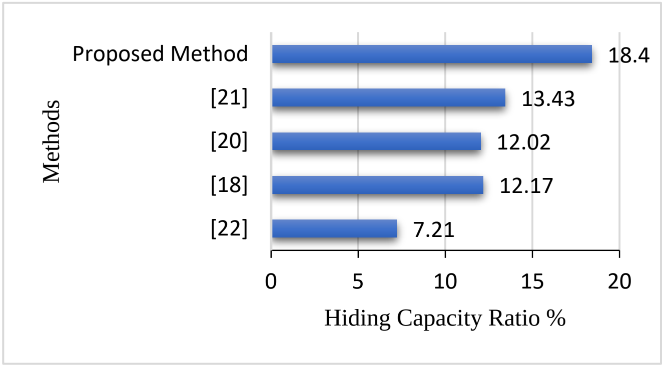

Fig. 13. Capacity of the proposed method compared with related studies

VI. CONCLUSION

This paper proposes a new technique of text steganography using Multilayer encoding with FPE and Huffman coding. The use of UCs shows a significant invisibility due to high imperceptibility after embedding the secret data into the cover text. Each Unicode character represents three bits of data. Before hiding secret data, the proposed scheme minimized the secret data size by applying multilayer encoding and Huffman compression. FPE is applied to secret data to achieve the encryption objective without increasing the size of the secret data. Results show that the proposed method has demonstrated significant improvement when compared with previous studies.

VII. FUTURE WORK

In future work, enhancement can be made by proposing a new method to solve the sequential pattern in the embedding process by introducing a randomization concept. Therefore, the security level will be increased without using any encryption techniques that require sharing of keys between participants. In addition, increasing the number of bits in UC mapping representation will result in increasing the amount of capacity in cover text.

ACKNOWLEDGMENT

The authors would like to acknowledge Universiti Kebangsaan Malaysia for the Research University Grant with code: TAP-K011439 to support this project.

REFERENCES

[1] 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., 2020.

[2] M. A. Majeed, R. Sulaiman, Z. Shukur, and M. K. Hasan, “A review on text steganography techniques,” Mathematics, vol. 9, no. 21, p. 2829, 2021.

[3] Ibrahim, A. H., & Alturki, A. S. (2020). Computational Analysis of Arabic Cursive Steganography using Complex Edge Detection Techniques. International Journal of Advanced Computer Science and Applications, 11(9).

[4] Shehab, D. A., & Alhaddad, M. J. (2022). Comprehensive Survey of Multimedia Steganalysis: Techniques, Evaluations, and Trends in Future Research. Symmetry, 14(1), 117.

[5] Almayyahi, A. A., Sulaiman, R., Qamar, F., & Hamzah, A. E. (2020). High-security image steganography technique using XNOR operation and fibonacci algorithm. International Journal of Advanced Computer Science and Applications, 11(10).

[6] Yang, Z., Xiang, L., Zhang, S., Sun, X., & Huang, Y. (2021). Linguistic generative steganography with enhanced cognitive-imperceptibility. IEEE Signal Processing Letters, 28, 409-413.

[7] Majeed, M. A., & Sulaiman, R. (2015). AN IMPROVED LSB IMAGE STEGANOGRAPHY TECHNIQUE USING BIT-INVERSE IN 24 BIT COLOUR IMAGE. Journal of Theoretical & Applied Information Technology, 80(2).

[8] Salah, A. H., Hadwan, M., Aqlan, A., Albazel, M., Alqasemi, F., & Al-Sanabani, M. (2021, August). A Survey on Different Arabic Text Steganography Techniques. In 2021 1st International Conference on Emerging Smart Technologies and Applications (eSmarTA) (pp. 1-8). IEEE.

[9] Kumar, R., & Yadav, A. K. (2021). Development of Novel Algorithm for Data Hiding on Mobile Application. International Journal of Advanced Computer Science and Applications, 12(8).

[10] Baziyad, M., Rabie, T., & Kamel, I. (2018, November). Extending steganography payload capacity using the L ab color space. In 2018 International conference on innovations in information technology (IIT) (pp. 1-6). IEEE.

[11] Wahab, O. F. A., Khalaf, A. A., Hussein, A. I., & Hamed, H. F. (2021). Hiding data using efficient combination of RSA cryptography, and compression steganography techniques. IEEE Access, 9, 31805-31815.

[12] B. Khosravi, B. Khosravi, B. Khosravi, and K. Nazarkardeh, “A new method for pdf steganography in justified texts,” J. Inf. Secur. Appl., vol. 45, pp. 61–70, 2019.

[13] L. Xiang, W. Wu, X. Li, and C. Yang, “A linguistic steganography based on word indexing compression and candidate selection,” Multimed. Tools Appl., vol. 77, no. 21, pp. 28969–28989, 2018.

[14] R. Kumar, A. Malik, S. Singh, and S. Chand, “A high capacity email based text steganography scheme using Huffman compression,” in 2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN), 2016, pp. 53–56.

[15] Fateh, M., & Rezvani, M. (2021). An email-based high capacity text steganography using repeating characters. International Journal of Computers and Applications, 43(3), 226-232.

[16] 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., 2020.

[17] D. Bhat, V. Krithi, K. N. Manjunath, S. Prabhu, and A. Renuka, “Information hiding through dynamic text steganography and cryptography: computing and informatics,” in 2017 international conference on advances in computing, communications and informatics (ICACCI), 2017, pp. 1826–1831.

[18] G. Maji and S. Mandal, “A forward email based high capacity text steganography technique using a randomized and indexed word dictionary,” Multimed. Tools Appl., vol. 79, no. 35, pp. 26549–26569, 2020.

[19] 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, 2017.

[20] 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 Advances in Intelligent Systems and Computing, vol. 843, 2019.

[21] 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, 2017.

[22] Kumar, R., Malik, A., Singh, S., & Chand, S. (2016, February). A high capacity email based text steganography scheme using Huffman compression. In 2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 53-56). IEEE.

[23] Gupta, S., Jain, S., & Agarwal, M. (2018, January). Ensuring data security in databases using format preserving encryption. In 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 1-5). IEEE.

[24] Pérez-Resa, A., Garcia-Bosque, M., Sánchez-Azqueta, C., & Celma, S. (2020). A new method for format preserving encryption in high-data rate communications. IEEE Access, 8, 21003-21016.

[25] 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.

[26] E. Brier, T. Peyrin, and J. Stern, “BPS: a format-preserving encryption proposal,” Submiss. to NIST, 2010.

[27] Cui, B., Zhang, B., & Wang, K. (2017, July). A data masking scheme for sensitive big data based on format-preserving encryption. In 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC) (Vol. 1, pp. 518-524). IEEE.

[28] Oh, I., Kim, T., Yim, K., & Lee, S. Y. (2019). A novel message-preserving scheme with format-preserving encryption for connected cars in multi-access edge computing. Sensors, 19(18), 3869.

[29] B. Carpentieri, A. Castiglione, A. De Santis, F. Palmieri, and R. Pizzolante, “Compression-based steganography,” Concurr. Comput. Pract. Exp., vol. 32, no. 8, p. e5322, 2020.