Indian Journal of Science and Technology, Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/87787, December 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Robust Digital Text Watermarking Algorithm based on Unicode Extended Characters

Nasr addin Ahmed Salem Al-maweri*, Wan Azizun Wan Adnan, Abdul Rahman Ramli, Khairulmizam Samsudin and Sharifah Mumtazah Syed Ahmad Abdul Rahman

Department of Computer and Communication Systems Engineering, Faculty of Engineering, Universiti Putra Malaysia, 43400, Malaysia; senassr_maweri@yahoo.com, wawa@upm.edu.my, arr@upm.edu.my, khairulmizam@upm.edu.my, s_mumtazah@upm.edu.my

Abstract

Objectives: A new text watermarking algorithm is proposed in this paper to protect documents from malicious attacks. **Methods:** In this paper, a novel digital text watermarking algorithm is developed based on Unicode extended characters. The algorithm is implemented to encompass watermark generation, embedding and extraction components. Predefined encoding tables are constructed to achieve the embedding while scrambling mechanism is proposed in generation and extraction to secure the watermarks. **Findings:** To evaluate the algorithm, it was tested, using ten different text samples, under various attacks such as conversion, reformatting, copying, insertions and deletions. The proposed algorithm has attained high level of imperceptibility with PSNR between 63.15 and 70.88 and SIM between 99.93% and 99.97%. Evaluating the robustness of the algorithm proved that it resists most of the attacks with high detection accuracy reached 100% in most cases. It has also achieved improved capacity with 2 bits/word and increased security level compared to the previous works. **Application/Improvements:** The proposed algorithm has added a new value to the information security field and can be used for documents protection from various malicious aspects.

Keywords: Extended Characters, Robustness, Scrambling, Text Watermarking, Unicode

1. Introduction

Information security in the digital world has become an important issue in the last few years due to the large volume of information exchanged over the world wide network. Information exchanged can be in a variety of forms such as audio, video, images and text. Threats, malicious attacks, illegal usage and violations being introduced to the exchanged information have brought a big challenge to the information security research area. Usually, the text remains the most transmitted media compared to the other forms. Because of that, it is the most threatened dominant to such illegal acts. For example, text documents can be illegally copied, tampered, redistributed, reproduced, leaked, or exposed to copyright and authentication violations. The daily illegal acts on text documents have put a new research direction for information security specialists.

Digital watermarking is one of the commonly used technologies to provide a protection to text document from such violations. In digital watermarking, additional

*Author for correspondence

information is inserted to the original content of the text. This information will be accompanying the text in a visible or an invisible way to be used as a proof of the originality of the document. The embedded information is known as watermark. In general, watermarking algorithms are designed to have three stages namely, watermark generation, watermark embedding and watermark extraction.

In the last few years, researchers have increasingly paid more attention to find new robust watermarking techniques for the text documents. However, the available techniques are still lacking the required robustness as well as there is still a need for a technique that enhances or attains some trade-off between imperceptibility, robustness and capacity. That is due to the difficulty of designing watermarking algorithms for text, since text has limited features compared to other media such as images and videos. Consequently, a new imperceptible, reliable, robust and secure as well as text watermarking algorithm which provides effective capacity is needed to solve the current text documents protection issues.

This paper is organized as follows: Section 2 reviews the recent watermarking algorithms for text documents. The 3rd Section defines the Unicode extended characters which has been utilized in the proposed work. The 4th Section presents in details the proposed text watermarking algorithm.

The 5th Section describes the results obtained from validating and evaluating the proposed algorithm. Finally, the 6th Section concludes the paper.

Research in digital text watermarking has taken various directions according to the applications types' requirement such as copyright protection, tampering protection, distribution control, and authentication and broadcast monitoring. According to this, the researchers have come up with different text watermarking techniques in trying to fulfill the emerged requirements.

Generally, the available digital text watermarking algorithms insert the watermarks to the text based on modifying the text to make it in a different status compared to the original text. The watermarks can be detected according to the modifications from the watermarked text. The available text watermarking algorithms can be categorized according to the method used during embedding. The following paragraphs will elaborated the available works according to this criterion.

Some algorithms use the format and structure properties of the text document such as line shifting[1], word shifting[2] or text features[3]. On the other hand, natural language watermarking modifies the text either in a semantic or syntactic way[4]. Some other techniques deal with the text document as an image and use image watermarking technique either by watermarking the text image in the spatial domain or in the wavelet domain. Other algorithms proposed using the text features and properties to generate a watermark key and register it with a trusted third party[5].

Recently, a new approach implemented in Microsoft Word document was proposed by[3]. For the purpose of embedding the watermark, two properties of the text were employed, Langauge ID and No proofing. By changing the values of one or both properties, 0 or 1 is considered embedded in the text. This algorithm has shown a high imperceptibility. However payload capacity was 0.5 bit/char. Moreover, changing the used properties into default values in word document will destroy the watermark data. Author in[6] introduced another approach to watermark text on web pages. In this method, the watermark insertion to the text was executed by using the tags in the web page source after converting it the watermark to hexadecimal form. In this algorithm a high imperceptibility is achieved. But, the watermarks are easy to lose if the source code was modified. Furthermore, watermark data security was neglected in this method. Author in[7] developed an algorithm to watermark web pages text as well. This algorithm extracts the features of the text by applying natural language syntactic and semantic analysis. Then the watermark is generated from the resulted analysis. The generated watermark is hashed then inserted to the text by adding white spaces between words. The watermark might be more robust in web pages by using white spaces. Using this algorithm in text documents will lack good robustness. In addition, the disadvantage of text abnormality with additional spaces will affect the imperceptibly. Another data hiding method called UniSpaCh that utilized inter-word spaces was earlier proposed[2]. This method was developed for Microsoft word documents. They used a combination of Unicode spaces and normal spaces such as six-per-em, hair space, thin space and punctuation to hide two bits from the watermark bit stream. This approach increases the

spaces between characters and paragraphs which will defiantly affect the imperceptibility level. It achieves a good capacity of 2 bits/word. But, modifications to the text easily affect the watermark detection accuracy.

Author in[8] developed a text watermarking algorithm which used the natural language perspective again. In this algorithm, the watermark key was generated by using the grammatical rules. The author's ID, modal verbs, pronouns and the count of conjunctions were combined together to create the watermark key. This watermark is registered with a Certifying authority for later usage. This method has good imperceptibility compared to other natural language watermarking algorithms, but lower robustness. Another natural language watermarking algorithm was proposed by[9]. In their proposed work, the algorithm was developed to watermark German text. For the purpose of embedding the watermark, syntactic transformation, conjunctions modulation, negations of words and lexical transformation (modifying words that have repeated letters) were utilized. The main merit of this method was its adaptability to different languages such as English, Spanish or French. But, this approach has the same disadvantages of natural language watermarking such as semantic drops and low robustness to content modifications. Author in[10] implemented a reversible watermarking method where the text can be recovered to the original state upon extraction for Chinese text. They suggested using synonyms substitution to embed the watermark information. Context collation degree of sentences and words were calculated and used for the purpose of decreasing the ambiguity and drops of content semantics. The main advantage of this method is reversing the watermarked text into its original status before extraction. Another work that was previously enhanced the avoidance of semantic drops was proposed by[11]. This method was implemented using some morpho-syntactic tools to build the syntax tree. The embedding of the watermark was achieved by altering the adverbs places and verbs replacement. This method has a low capacity where it was only able to hide 0.5 to 1 bit/sentence and limited to Turkish language. Natural language watermarking was first proposed by[12]. After generating a syntax tree from the text, the embedding was executed. Before that a secret key was constructed from the text then inserted to syntax tree by modifying transformations in the generated tree.

A new text watermarking perspective was introduced recently to avoid direct amendments to the text itself resulting in high imperceptible algorithms. These techniques are known as zero watermarking. Author in[13] used Term Frequency–Inverse Document Frequency (TF-IDF) tool in their method to analyze the text features, particularly frequency of words. The watermark key was generated from the features then hashed using MD5. The hashed watermark then was registered in a certifying authority. A zero watermarking algorithm was proposed by[14]. In this algorithm, the embedding of the watermark was implemented by generating watermark key as in the other zero watermarking algorithms. The count of letters in two words before and after a predefined keyword was used. The resulted numbers through the whole text were concatenated in one string to construct the watermark to register it with a certifying authority. Author in[15] implemented another zero watermarking method. This time the embedding of the watermark was implemented by choosing the words that had more than four characters. The first character in each of the chosen word was appended to the generated watermark key. The resulted string was registered with certifying authority. Author in[16] proposed a method that utilizes the occurrences of letters in each word by choosing the first letter from the word that has this letter more than once. A pattern was generated from those letters to build the watermark key and register it with a trusted third party. Quite recently author in[17] developed a zero watermarking method based on Markov model. Text was analyzed using the Markov model of order three to generate a pattern from each three unique consecutive letters. The output from Markov model will be in a form of sequence numbers. This sequence is hashed using MD5 and then registered with a third party for the extraction purpose. Author in[18] proposed another algorithm based on zero watermarking idea. They used double letters list combined with a predefined image-plus-text watermark to generate the key watermark. The image-plus-text watermark was mapped to a text before mixing it with the double letters list. The combination key was then registered with certifying authority.

The above zero watermarking algorithms have introduced some improvement in imperceptibility since the watermark is logically embedded. However, it remains limited to only a few applications. Amendments in the text affect the features and consequently the robustness and leads to the incorrect watermark detection.

The previous discussed algorithms deal with text as text. At the same time, some algorithms have been proposed to treat text as binary images. Based on that, known image watermarking techniques can be applied to text binary images as well. Author in[19] developed an algorithm that utilized the four bits of RGB color in every letter benefiting from the inability of the human visual system to detect slight changes in the colors. The watermark data were embedded in a binary form by changing the four bits values according to the watermark input bits. Author in[20] reported the development of a method that used the spatial domain to watermark text documents. In their method, the insignificant pixels in the background of the text were utilized to embed the watermark data. Another method was proposed in[21] that used the known word-shift technique. By shifting the word, inter-word spaces was amended after constructing a sin wave watermark from the original inter-word space. Author in[22] proposed a method of watermarking travelling text documents over internet. The text document was represented in a spread spectrum signal. The watermark was generated from the text features and added to Pseudo Random Number Generator (PRNG), then spread it over the original signal. The watermarked text signal was used in the receiver side to extract the watermark by subtracting Pseudo Random Number (PRN) from the text signal.

Treating the text documents as images has its drawback of degrading the text quality as well as the low resistance to image manipulations.

Previous works in text watermarking algorithms are lacking in satisfying robustness and imperceptibility requirement which is considered as a hard task. Dealing with plain text makes finding a technique to hide the watermark data in text difficult since text properties are limited. Besides, adding or modifying text for the purpose of hiding other information on the text will definitely affect the original text and would lead to a weak performance. Most algorithms discussed in the previous section, suffer from low imperceptibility, robustness and capacity.

2. Unicode Extended Characters

In this work, a new technique is proposed for text watermarking. The proposed technique employs Unicode Extended Characters as the object used to hide the watermark information in the plain text. Microsoft word processor and most of the text editors support Unicode extended characters. The usual typed text in any text editor utilizes basic Latin characters, or commonly called ASCII characters with 8 bits code. Furthermore, most operating systems include the Unicode standard[23].

The idea of utilizing Unicode extended characters as the basis of the developed text watermarking embedding operation is because Microsoft Word supports these characteristics. In addition, the presence of same alphabetical letter in different code numbers was the courage for inventing the idea of employing the extended characters to hide the watermark bits. For example, letter 'A' has ASCII code of '65' and Unicode of '1040'. Simple coding tables are designed to choose the suitable ASCII and Unicode extended characters to be utilized by the embedding operation. Unicode extended characters has provided a robust, high imperceptible and enough capacity metrics to the developed text watermarking algorithm.

3. Proposed Algorithm

3.1 Architecture

The proposed text watermarking algorithm is designed to achieve the purpose of inserting the watermark bits stream into the Word document text. The text watermarking algorithm components are described in Figure 1. Text watermarking algorithm consists of three main components, namely, watermark generation, watermark embedding and watermark extraction. Watermark Generation component is responsible for generating the array elements of the watermark data received from the author of the document. The generation component will convert the watermark into stream of binary bits then encode or scramble the watermark using a 5 digits key

inserted by the user. The scrambling mechanism will be explained in section B.

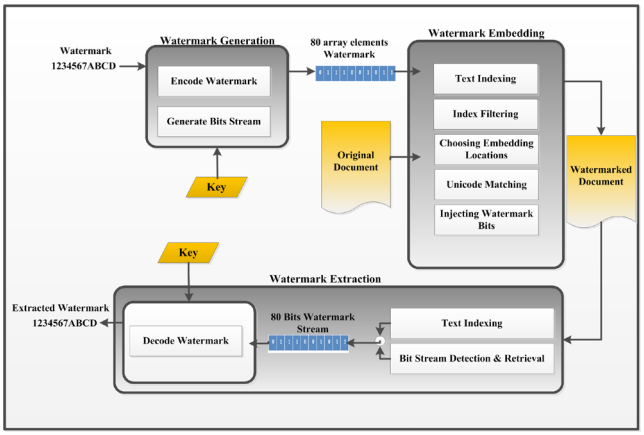

Figure 1. Proposed text watermarking algorithm architecture.

Currently, the watermark length is fixed to 80 bits; however the algorithm is designed to be flexible to longer or shorter length. The output from the watermark generation will be reconstructed as array elements and then will be used by the embedding component. The embedding component is designed to insert the binary watermark elements into the original text document. This component runs by executing the following operations which include: Text indexing, index filtering, choosing embedding locations, Unicode characters matching and injecting watermark bits. The output from the embedding component will be the watermarked document that contains a hidden stream of bits. The third component is called watermark extraction. This component is designed to detect and retrieve the watermark bits from the watermarked document. The extraction is achieved by running two operations, text indexing and bits stream detection and retrieval. The output from these operations is in a form of bits stream. Then the watermark bits stream is decoded by using the same scrambling mechanism and same key used during the embedding to recover the original embedded watermark.

3.2 Watermark Generation and Scrambling

As a pre-processing for the embedding phase, the first component to run the proposed algorithm is watermark generation. First, the received watermark data from the user are converted to binary bits which are later stored as array elements. As mentioned previously, the watermark length in the developed algorithm was tested under 80 bits long. For protecting the watermark information, securing the watermark data requires using a key that contains five numerical digits. A method is proposed to scramble the watermark data using the generated key. Figure 2 depicts the text watermarking generation process.

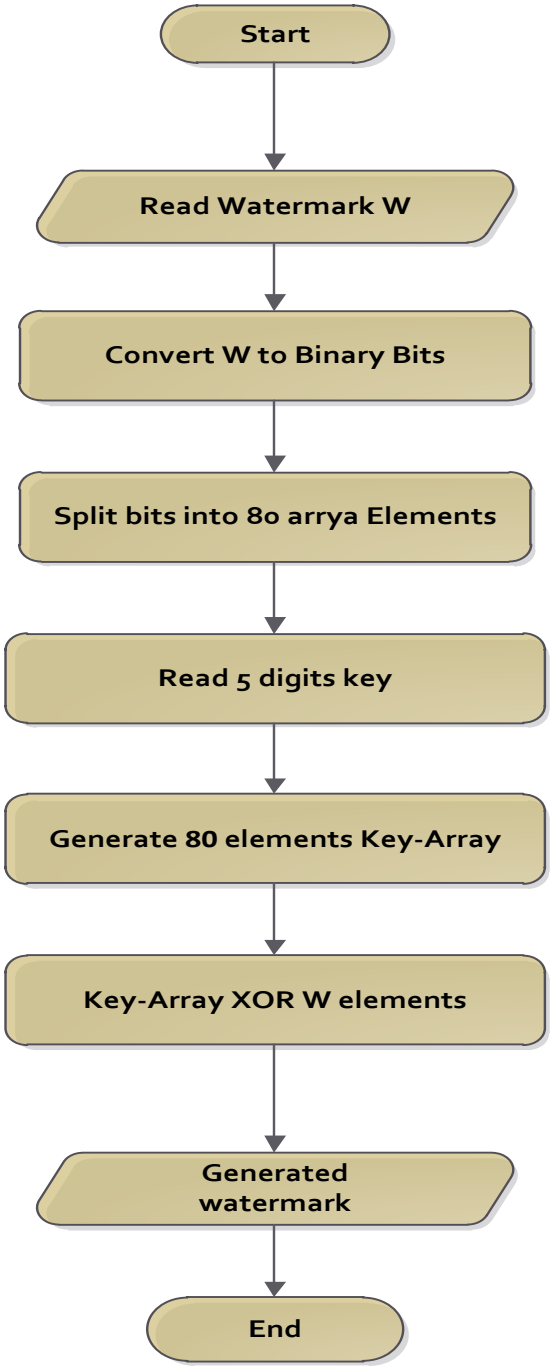

Figure 2. Watermark generation.

The generation of the key stream and the scrambled watermark take the following steps:

- Read the key.
- Convert the key from integer to binary form.
- Expand the resulted binary bit stream to cover the whole 80 elements of watermark Array.
- XOR the generated 80 elements of the key with the watermark array.
- Send the encoded watermark array after the XOR operation to the embedding function.

3.3 Watermark Embedding

Watermark embedding process shows the method of inserting the watermark bits into the text document to be protected. The main purpose of the embedding component in the proposed algorithm is to embed the watermark elements generated in the previous step to the text document. The embedding process detailed implementation is explained in the following sections. Figure 3 describes the embedding process.

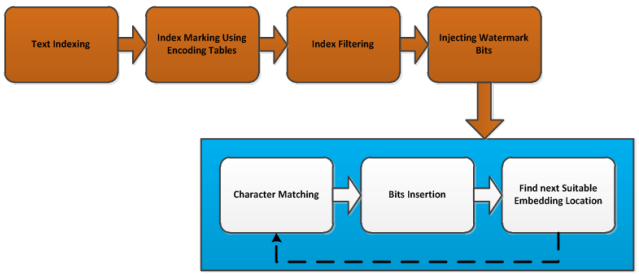

Figure 3. Watermark embedding process.

3.3.1 Encoding Tables

In the proposed algorithm, binary numbers are considered to represent the watermark information needed to be inserted in the text document. According to this, the watermark unit will be either 0 or 1. To achieve embedding 0 or 1 in the plain text, available similar Unicode characters are chosen to represent either 0 or 1. Two encoding tables are constructed to separate the available characters into two groups. First group means 0's are embedded and the second group means 1's are embedded. These tables are named Zero-Encoding Table and One-Encoding Table respectively. Each table contains list of ASCII characters and their codes. This list is associated with their similar Unicode extended characters and their codes which are used for the embedding purpose. Tables 1 and 2 show how those tables are constructed. The tables show the chosen ASCII characters to carry the watermark bits including (small and capital letters). The associated ASCII codes and Unicode characters watermarking code are listed in column 2 and 3 from each table where Table 1 is constructed for Zero-Encoding and Table 2 is constructed for One-Encoding.

Table 1. Zero-encoding table

Character	ASCII Code (Decimal)	Watermarking-Code (Decimal)
a	97	1072
c	99	1089
e	101	1077
h	104	1211
i	105	1110
A	65	1040
B	66	914
C	67	1057
E	69	917
H	72	919
I	73	921
J	74	1302
K	75	922
M	77	924

Table 2. One-encoding table

Character	ASCII Code (Decimal)	Watermarking-Code (Decimal)
j	106	1112
o	111	1086
p	112	1088
s	115	1109
x	120	1093
y	121	1091
N	78	925
O	79	927
P	80	929
S	83	1029
T	84	932
X	88	935
Y	89	933
Z	90	918

3.3.2 Text Indexing

During the implementation of the algorithm, processing the text document (Word document) directly from the hard-disk has shown a large delay on executing the watermarking process. Currently, many search engines use the indexing technique for the purpose of retrieving information from the internet. Using such technique will enhance the performance and speed up the process of getting the search result. In indexing, instead of searching for the keyword directly in the documents, a prior processing for these documents is executed to index the documents text in a specific way which facilitates the search later. The keyword is then searched against the indexes, instead of dealing with the keyword. String matching is considered one of the most trivial topics in information retrieval. Indexing the string itself has taken the same way used in searching by index. This technique is utilized in the proposed text watermarking algorithm to enhance the embedding computation time by indexing the document text as a prior processing before really dealing with the watermark embedding process. This is achieved by assigning each character in the document a number that refers that refers to its index. The index values and characters are stored in arrays to simplify the access. Those indexing arrays are used during the embedding and extraction instead of the text document itself. Figure 4 depicts the processes of how indexing technique was implemented in the embedding operation.

3.3.3 Index Filtering

As a result from the indexing process execution, two indexing arrays are created with the same size of the general indexing array, one for zero-allowed indexes and the other for one-allowed indexes. After filling both arrays, some elements will be having no values. As described in the encoding tables, these unknown values are obtained because that there are some characters, in the text document, not considered to allow neither 0 nor 1. To enhance the performance of the algorithm and to avoid missing up with indexes, a filtering process is followed to purify the indexing arrays to have only the embeddable indexes values in both arrays. Figure 4 shows in the second part how the filtering process is implemented. Simply, new arrays called filtered-indexes-zero and filtered-indexes-one are created to store the values of indexes above than zero where zero refers to unknown index in the previous created arrays. To filter the indexing arrays, the marked character index is read, if it is marked as zero-allowed and the index is greater than 0, append the value in filtered-indexes-zero. Otherwise, check if it is marked as one-allowed and the index is greater than zero append the index to filtered-indexes-one.

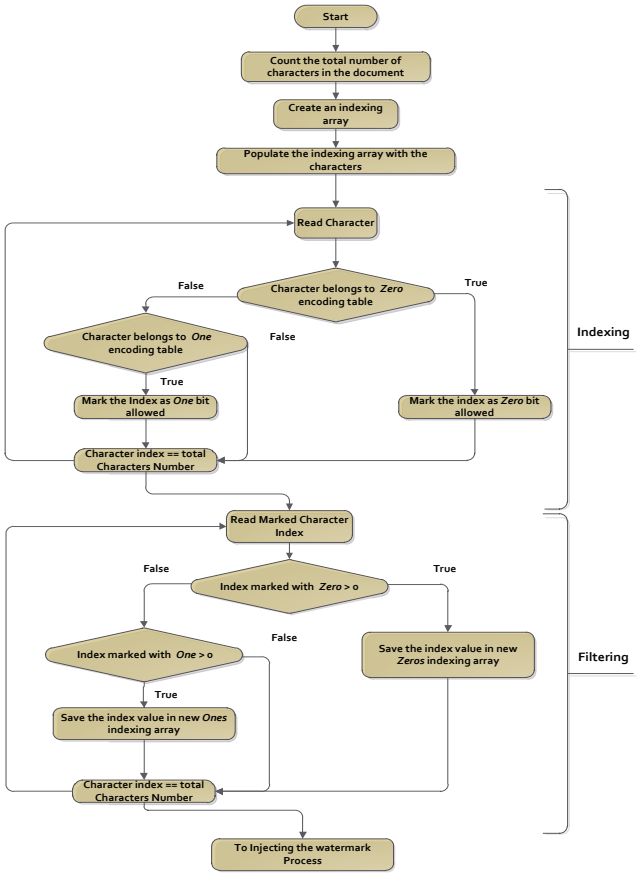

Figure 4. Text indexing and filtering.

3.3.4 Watermark Bits Injecting

After indexing the text characters and executing the filtering process, filtered-indexes-zero and filtered-indexes-one are used to during bits injection at the main text. Figure 5 shows the steps of how the watermark bits injecting operation was implemented. To implement the bits injecting process which is considered as the most important part in the embedding operation, a pointer to the character index is created to move through the text characters and inject the bits one by one. The pointer points to nothing in the beginning. To activate the pointer, the first bit in the watermark array is checked, if it is 0, the pointer is moved to point at the first character index marked as *zero-allowed* in the associated indexing array. Otherwise it is moved to point to the first character index marked as *one-allowed* in the associated indexing array.

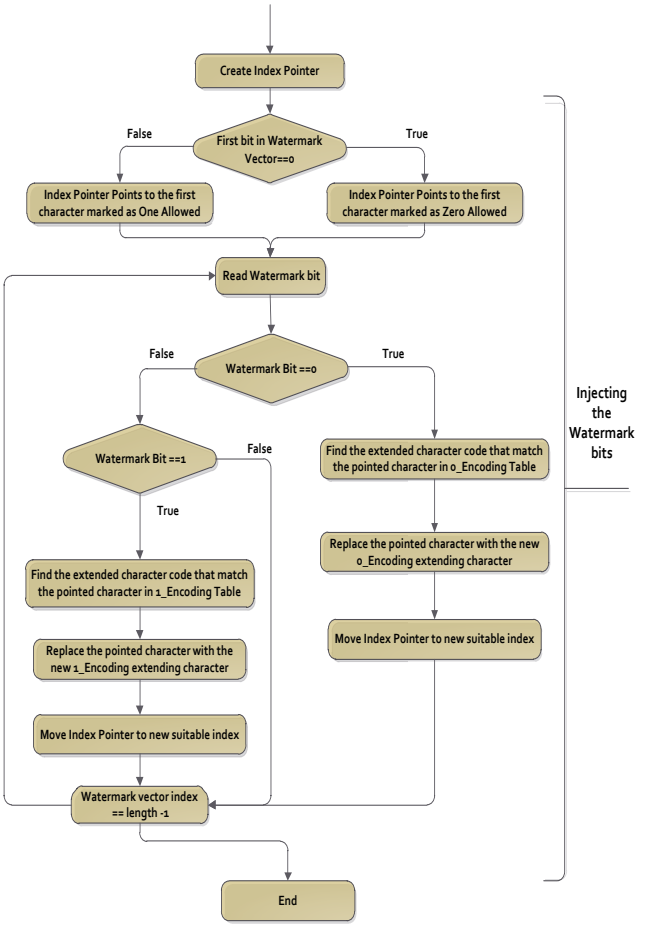

Figure 5. Watermark bits injection.

The details of each step involved in bits injecting is as follows:

Step 1: The bit value in the pointed index is read and checked. If the bit value is 0, the process of Character Matching (refer to Section 6) is run on zero-encoding table to return the code of the extended Unicode character that matches the ASCII indexed character.

Step 2: The ASCII character is replaced in the text with the returned extended Unicode character using the returned code.

Step 3: Find the next suitable position according to the value of the next bit and move the pointer to that location. This is achieved by 'find embedding location' process which will be discussed in Section 5.

Step 4: Otherwise, if the bit to be inserted is 1; previous steps are repeated but now with considering the value 1 and one-encoding table instead. The process of Character Matching is run on one-encoding table to return the code of the extended Unicode character that matches the ASCII indexed character. After that, the ASCII character is replaced in the text with the returned extended Unicode character using the returned code.

Step 5: Find the next suitable character position according to the value of the next bit and move the pointer to that location. This is achieved as well by 'find embedding location' process.

The above steps are repeated until it achieves 80 bits injection in the text characters.

3.3.5 *Finding Embedding Locations*

During the injection of the watermark bits, there is a need for preparing the next embeddable and suitable character for injecting the next bit. This preparation is according to the value of the next bit that will be injected. Suppose that the pointer is in the current bit which is for example 0, and the next bit to be injected is 1. In this case, the injection process needs to know where to inject the coming 1 bit, due to the probability of the presence of 0 embeddable characters follow the current replaced character. According to this some characters need to be skipped and the pointer must move to the first character index that allows injecting a bit of value 1. Figure 6 explains how the next index is prepared.

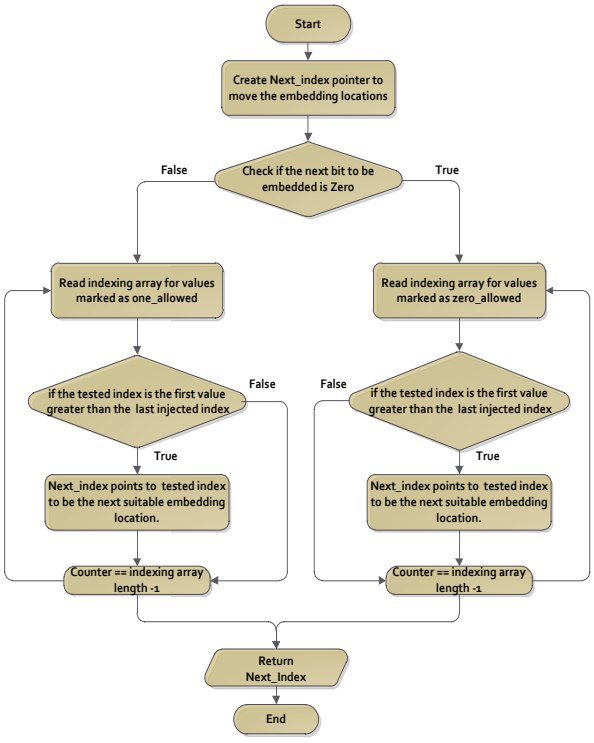

Figure 6. Finding embedding location.

3.3.6 Character Matching

In the embedding operation, before replacing the ASCII characters in the main text, the injecting process needs to know which character matches the character to be replaced in the encoding tables. This matching is executed after deciding whether the bit to be injected is 0 or 1. The function of character matching returns the code of the Unicode extended character that matches the ASCII character from the encoding tables according to the value of the current bit needed to be inserted.

3.4 Watermark Extraction

The extraction component in the developed algorithm aims to extract the watermark data from the watermarked text document. The following sections will explain in details how the extraction component is implemented. Figure 7 describes the watermark extraction.

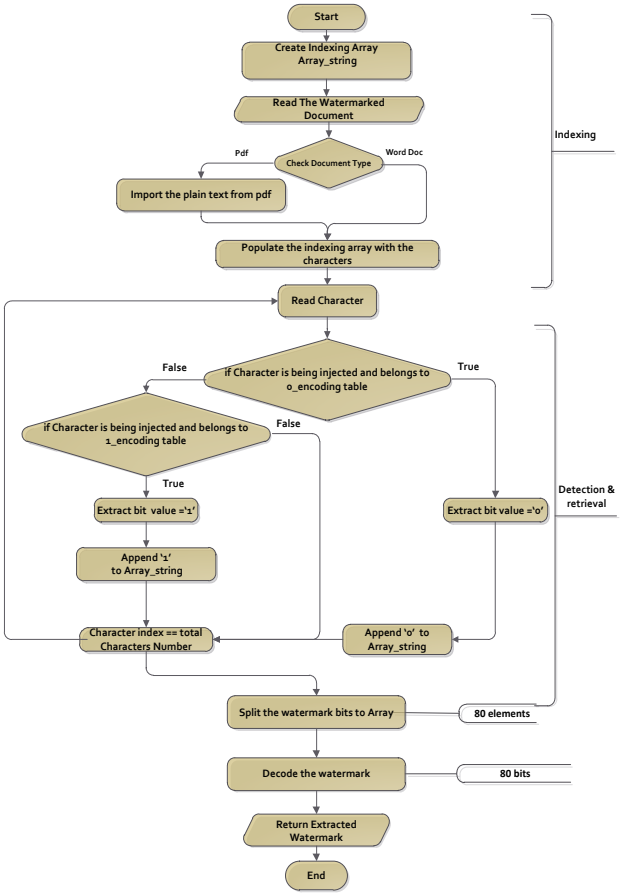

Figure 7. Watermark extraction.

3.4.1 Text Indexing

As text indexing is used in the embedding operation, it will be used again in the extraction operation to enhance the computation time and to speed up the process of extracting the watermark from the text document. That is because dealing with text from the document in a direct way and looping over it definitely showed much more consumption time. Figure 7 shows how the indexing process is integrated to the extraction component.

After the document is loaded, the extraction function will start with recognizing if the document is a word document or pdf document. The same indexing process will be followed in both situations. The only difference is in the case of pdf document, before indexing the text, the text itself is imported from the pdf using a free library called pdf box. This library has a function called PDF Stripper which retrieves the content of a pdf file as it is. After the text is imported from the pdf, the indexing process can begin.

In the text extraction function the indexing process is made simple by just creating one indexing array, reading the text character and populating the indexing array to be used during the bits detection and retrieval.

3.4.2 Watermark Bits Detection and Retrieval

After the watermarked text is being indexed in an indexing array, the time has come to detect and retrieve the watermark bits from the text. Figure 7 shows how the detection of the injected characters and retrieval of the bits from these characters work.

The detailed explanation of the steps involve in bits retrieval is as follows:

Step 1: the extraction function iterates over the indexed characters one by one. Each time, the character is checked if it has a Unicode extended character code. This is to detect that the character is carrying a watermark bit or not.

Step 2: retrieve either 1 or 0 to construct the watermark Array-string variable. To do that, the code of the character is read, and then checked. If the code belongs to Zero-encoding table, retrieve 0. Otherwise, check if the character code belongs to one-encoding table, retrieve 1.

Step 3: go to the next character until the 80 embedded bits have been extracted.

Figure 8 presents a sample of watermarked text and its associated watermark. The characters which highlighted are the chosen characters for inserting the bits according to the encoding tables.

3.4.3 Watermark Decoding

To recover the correct original watermarks the same key used by the algorithm to generate the scrambled watermark must be used again to decode the extracted watermark. Retrieving the decoded watermark involves repeating the steps in the generation process.

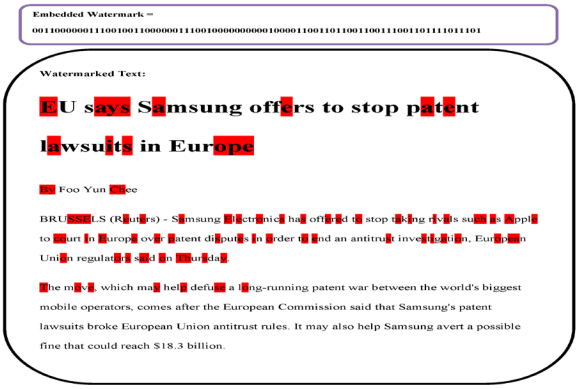

Figure 8. Sample of watermarked text.

4. Experimental Results and Discussion

To validate the performance of the proposed algorithm, ten text document samples from different websites such as Reuters, Official Governmental documents, News Papers, and Articles are utilized. These ten samples are input to the algorithm one by one. The algorithm then is run to embed ten different watermarks, each in one text document sample. The watermarked text document samples with the original associated watermarks are saved for the next evaluation process. Each original text document sample will have its own watermarked version. These two versions are later used to evaluate the imperceptibility as well as the capacity metrics using both PSNR (Peak Signal to Noise Ratio) and SIM (Similarity) according to the following steps.

In the first step the root mean squared error (RMSE) is calculated which is given by the following formula.

$$RMSE = \sqrt{\frac{1}{mxn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[O_{doc}(i,j) - W_{doc}(i,j)]^2} \quad (1)$$

Where mxn is the size of the image, (i,j) is the pixel location, O_{doc} is the original document image and W_{doc} is the watermarked image document. In the second step, according to the value resulted from the previous equation the PSNR is defined as

$$PSNR = 20\log_{10}\frac{MAX(O_{doc})}{RMSE} \quad (2)$$

Where $MAX(O_{doc})$ is the maximum pixel value in the document image. In this case it is equal to 255 since the text document was transformed into 8 bits image.

Then, the similarity percentage is found as follows:

$$SIM = \left(1 - \frac{RMSE}{MAX(O_{doc})} x100\right)\% \quad (3)$$

The watermarked version of each text document sample is attacked (Conversion, copying, reformatting, insertion or deletion) then utilized as input to the extraction component after applying the attack.

In conversion attack, the watermarked Word documents are converted to four different extensions such as pdf, html, rtf and odt, and then these files are used to extract the watermarks. In, copy attack, the text in the watermarked document, is exposed to copying to different editing environment such as Web, OpenOffice, Wordpad and NotePad, saved, and then the copied text is used in extraction. In reformatting attack, the layout, font, and text alignment are modified randomly. In insertion attack, different percentage of additional text was added to the watermarked text either in one place (localized) or in multiple places (dispersed). In deletion attack, the watermarked text is exposed to characters, words and phrases deletions either from one place (localized) or from multiple places (dispersed).

The extracted watermarks are then compared with the original watermarks in order to measure the robustness of the algorithm by computing the detection accuracy using hamming distance function. Hamming Distance (HD) measures the number of locations that differs between two bit streams. In other words, it defines the error rate between two bit strings. For Example

B1 = 01110101101
B2 = 11110110101
Hamming Distance (B1, B2) = 3

The resulted hamming distance value was then used to compute the Detection Accuracy Percentage which is given by:

$$Detection\ Accuracy = \left(\frac{N_{bits} - \boldsymbol{HD}}{N_{bits}} x100 \right)\% \qquad (4)$$

Where N_{bits} the number of watermark is bits and **HD** is the hamming distance.

The results obtain from evaluating the proposed algorithm is discussed in the next sub sections.

4.1 Imperceptibility

Peak Signal to Noise Ratio (PSNR) and Similarity Percentage (SIM) are used to ensure that a good perceptual quality and high similarity percentage have been achieved in the proposed text watermarking algorithm. PSNR acceptable value should be above 30[24] and it shows that the proposed algorithm has achieved high PSNR and SIM values as shown in Figure 9 and Figure 10. In the proposed algorithm, the obtained PSNR value is between 63.15 and 70.88 and the similarity is between 99.93% and 99.97% in all the samples. Whereas, UniSpaCh introduced too much deterioration to the original document quality and PSNR values are below 30 and similarity percentage between 88.96% to maximum 93.13%.

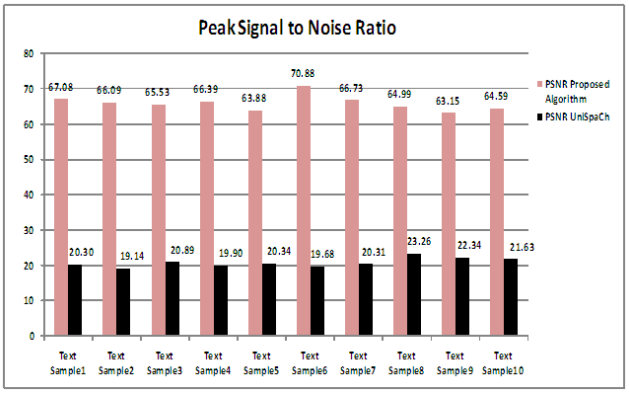

Figure 9. PSNR of proposed algorithm vs UniSpaCh.

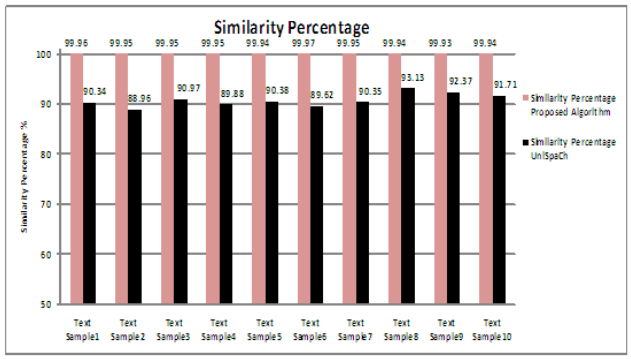

Figure 10. Similarity percentage of proposed algorithm vs UniSpaCh.

4.2 Robustness and Security

The robustness of the proposed algorithm is tested by utilizing 80 bits watermark and compared with UniSpaCh algorithm by exposing both algorithms to the same attacks. The detection accuracy is computed in each text sample. The experimental result shows that the proposed algorithm resists copying, conversion, reformatting attacks where the detection accuracy under these attacks is 100% as can be seen in Figures 11, 12 and 13. UniSpaCh attained similar results under these attacks; however it showed less robustness when the word document is converted to pdf file. In addition, the proposed algorithm is also tested for insertion and deletion attacks and shows a high level of detection accuracy compared to UniSpaCh algorithm as can be seen in Figures 14, 15, 16 and 17. The proposed algorithm survived both dispersed and localized insertion attacks, while UniSpaCh failed in most cases. The proposed algorithm has also showed better robustness in both kinds of deletion attack compared to UniSpaCh.

Speaking about watermarking security, one should differentiate between security and robustness. Where robustness refers to how the watermark resists attacks to persist in the watermarked text; while a secured watermark refers to how the watermark information are being unrevealed even if the watermark was extracted maliciously[25]. Therefore, it can be said, the proposed algorithm has considered the security part since the watermark embedding is executed using two keys, each one has five digits. Without the keys it is impossible to retrieve the right information for the watermark.

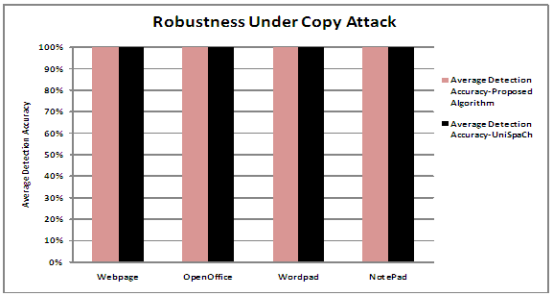

Figure 11. Robustness under copy attack of proposed algorithm vs UniSpaCh.

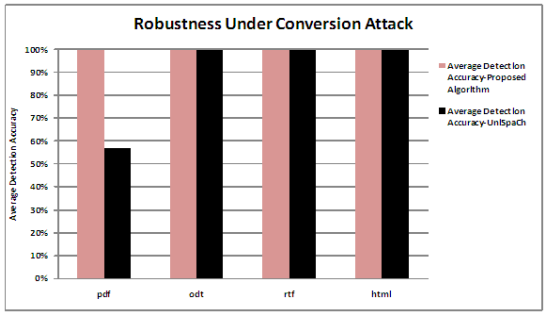

Figure 12. Robustness under conversion attack of proposed algorithm vs UniSpaCh.

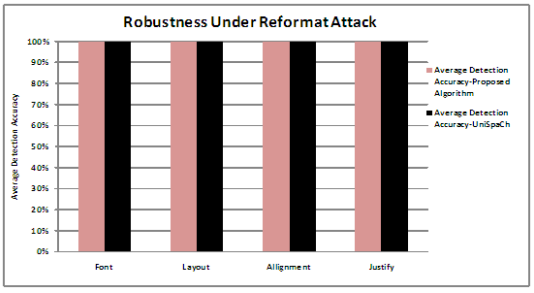

Figure 13. Robustness under reformat attack of proposed algorithm vs UniSpaCh.

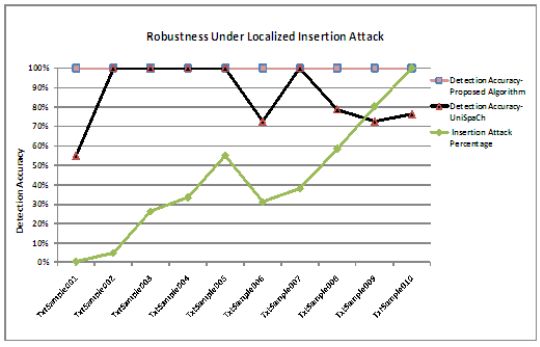

Figure 14. Robustness under localized insertion attack of proposed algorithm vs UniSpaCh.

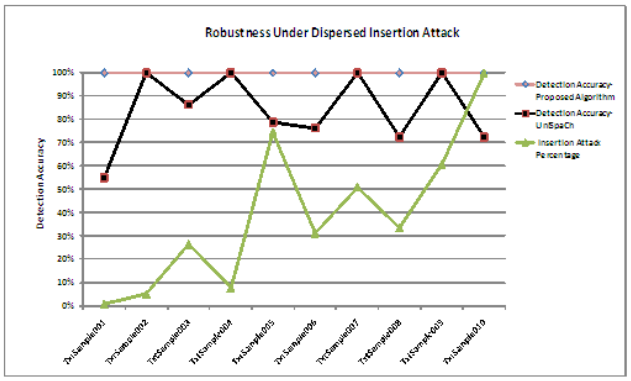

Figure 15. Robustness under dispersed insertion attack of proposed algorithm vs UniSpaCh.

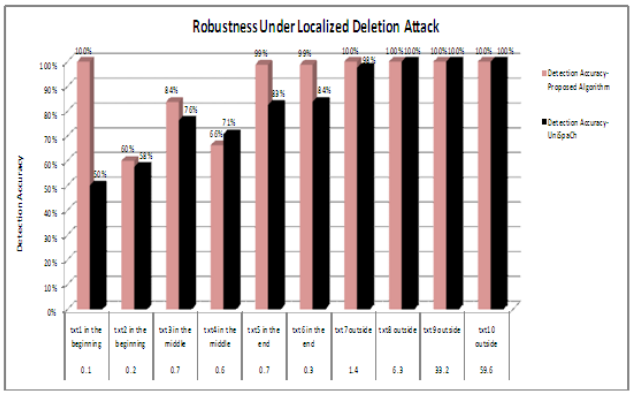

Figure 16. Robustness under localized deletion attack of proposed algorithm vs UniSpaCh.

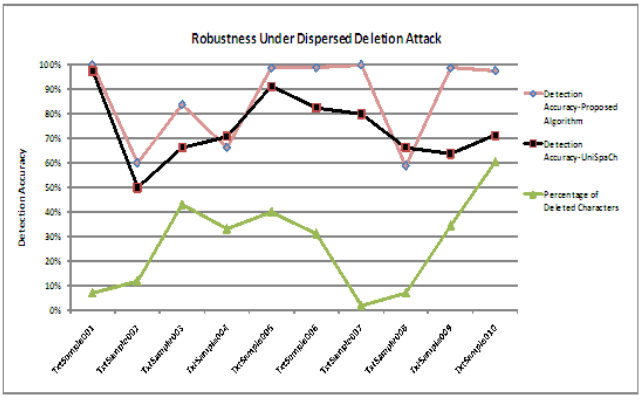

Figure 17. Robustness under dispersed deletion attack of proposed algorithm vs UniSpaCh.

4.3 Capacity

Evaluating the capacity of the proposed algorithm was reported in three factors payload capacity, granularity and file size. The algorithm has achieved a payload capacity

of about 2 bits/word where UniSpaCh attained the same value. The granularity has shown that the proposed algorithm is able to embed the watermark bits in reasonable number of character. Figure 18 shows the granularity of the proposed algorithm in comparison with UniSpaCh. Table 3 shows that the algorithm has introduced a low impact on the file size after watermarking the text documents.

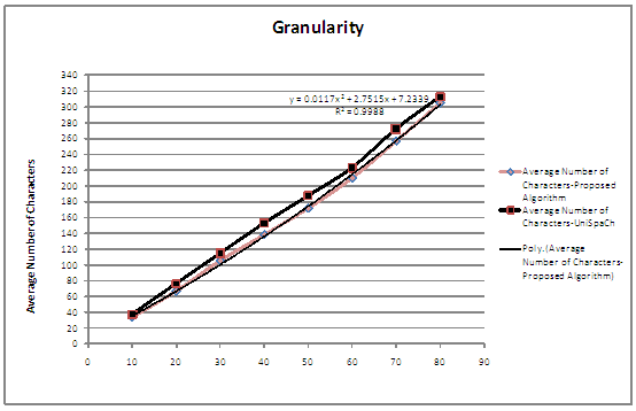

Figure 18. Granularity of the proposed algorithm of proposed algorithm vs UniSpaCh.

Table 3. File size impact of proposed algorithm vs UniSpaCh

Text Sample	Proposed Algorithm Size Increase%	UniSpaCh Size Increase%
1	4.07	4.62
2	3.04	3.37
3	2.83	3.72
4	3.11	11.47
5	3.77	3.70
6	5.13	3.21
7	5.23	2.86
8	3.20	4.83
9	9.50	9.65
10	3.18	3.13
Average	4.31%	5.06%

5. Conclusion

In this paper a novel and robust text watermarking algorithm has been proposed. The algorithm has been implemented using the Unicode extended characters as the object of hiding the watermark data. Zero-encoding table and One-encoding table are designed to achieve the embedding purpose. The proposed algorithm performance has been verified and compared with similar methods to ensure its imperceptibility, robustness and capacity. The developed algorithm provides a high imperceptible watermarking technique where it attains about 99.9% of similarity factor keeping the perceptual quality of the watermarked documents to have a high PSNR value above 63. Robustness evaluation proves that the proposed algorithm tolerates most of the possible attacks and able to extract the watermark with high accuracy. Capacity factors evaluation shows that the proposed algorithm has an acceptable watermarking capacity with a payload capacity of about 2 bits/word. Improving the proposed algorithm to recover re-ordering attack and investigating other Unicode properties are considered as interesting future research directions.

6. Acknowledgement

This research was supported by the Ministry of Higher Education Malaysia and Universiti Putra Malaysia under Exploratory Research Grant (ERGS).

7. References

1. Patel MM. Analytical study of line-shift text watermarking technique. International Journal of Computer Applications and Information Technology. 2012 Nov; 1(3):84–7.
2. Por LY, Wong KS, Chee KO. UniSpaCh: A text-based data hiding method using Unicode space characters. The Journal of Systems and Software. 2012; 85(5):1075–82.
3. Ruia X, Jinqiaob CXJS. A multiple watermarking algorithm for texts mixed Chinese and English. Procedia Computer Science. 2013; 17:844–51.
4. Singh P, Chadha RS. A survey of digital watermarking techniques, applications and attacks. International Journal of Engineering and Innovative Technology (IJEIT). 2013 Mar; 2(9):165–75.
5. Kaur H, Kaur ES. Text watermarking using techniques dct and dwt: A review. International Journal of Computer Application and Technology. 2014 Apr; 1(1):1–6.
6. Jaiswal RJ, Patil NN. Implementation of a new technique for web document protection using Unicode. IEEE 2013 International Conference on Information Communication and Embedded Systems (ICICES); 2013 Feb. p. 69–72.

7. Mir N. Copyright for web content using invisible text watermarking. Computers in Human Behaviour. 2014; 30:648–53.
8. Mali ML, Patil NN, Patil JB. Implementation of text watermarking technique using natural language watermarks. IEEE 2013 International Conference on Communication Systems and Network Technologies; Chennai. 2013 Feb 21–22.
9. Halvani O, Steinebach M, Wolf P, Zimmermann R. Natural language watermarking for German texts. IH&MMSec *'13* Proceedings of the first ACM workshop on Information hiding and multimedia security. 2013 Jun. p. 193–202.
10. Fei W, Tang X. Reversible text watermarking algorithm using prediction-error expansion method. International Conference on Computer, Networks and Communication Engineering (ICCNCE 2013); Atlantis Press. 2013 May.
11. Meral HM, Sankur B, Sumru A, Tunga, Sevinc E. Natural language watermarking via morphosyntactic alterations. Computer Speech and Language. 2009 Jan; 23:107–25.
12. Atallah MJ, Raskin V, Crogan M, Hempelmann C, Kerschbaum F, Mohamed D, Naik S. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. Proceedings of the 4th International Workshop on Information Hiding; Springer-Verlag. 2001. p. 185–99.
13. Kuang Q, Xu X. A new zero-watermarking scheme based on features extraction for authentication of text. Journal of Convergence Information Technology (JCIT). 2011Nov; 6(11):155–65.
14. Jalil Z, Mrirza AM, Sabir M. Content based zero-watermarking algorithm for authentication of text documents. International Journal of Computer Science and Information security. 2010 Feb; 7(2):212–7.
15. Jalil Z, Mrirza A, Jabeen H. Word length based zero-watermarking algorithm for tamper detection in text documents. IEEE 2010 2nd International Conference on Computer Engineering and Technology; Chengdu. 2010 Apr 16–18.
16. Kaur S, babbar G. A zero-watermarking algorithm on multiple occurrences of letters for text tampering detection. International Journal on Computer Science and Engineering (IJCSE). 2013; 5:294–301.
17. Ba-Alwi FM, Ghilan MM, Al-Wesabi FN. Content authentication of English text via internet using zero watermarking technique and Markov model. International Journal of Applied Information Systems (IJAIS). 2014; 7(1):25–36.
18. Rameshbabu P, Prasannakumar, Balachandrudu KE. Text watermarking using combined image and text. IJERT. 2013 Dec; 2(12):3812–8.
19. Du M, Zhao Q. Text watermarking algorithm based on human visual redundancy. Advanced in Information Sciences and Service Sciences. 2011; 3(5):1–7.
20. Puhan NB, Ho ATS, Sattar F. Erasable authentication watermarking in binary document images. IEEE Second International Conference on Innovative Computing, Information and Control; Kumamoto. 2007 Sep 5-7. p. 1–288.
21. Huang D, Yan H. Inter-word distance changes represented by sine waves for watermarking text images. IEEE Transactions on Circuits and Systems for Video Technology. 2001 Dec; 11(12):1237–45.
22. Qadir MA, Ahmad I. Digital text watermarking: Secure content delivery and data hiding in digital documents. E systems Magazine. 2006 Dec; 21(11):18–21.
23. Unicode Tutorials and Overviews, Unicode Std. Available from:http://www.unicode.org/standard/tutorial-info.html#TUS
24. Huynh-Thu Q, Ghanbari M. Scope of validity of PSNR in image, video quality assessment. IEEE Electronics Letters. 2008 Jun 19; 44(13):800–1.
25. Sujatha P, Devi R. An overview of digital watermarking with a performance analysis of wavelet families for image compression. Indian Journal of Science and Technology. 2015; 8(29):1–5.