Unified Inter-Letter Steganographic Algorithm, A Text-based Data Hiding Method

Ahmad Esmaeilkhah[1], Changiz Ghobadi[1], Javad Nourinia[1], and Maryam Majidzadeh[2]
[1]Electrical Engineering Department, Urmia University, Iran
[2]Department of Electrical and Computer Engineering, Technical and Vocational University, Iran

Abstract: *This paper funds a novel text-based steganographic algorithm with enhanced functionality with respect to the previously proposed methods, by careful selection of one of standard space characters, the introduced Inter-Letter Steganographic Method, or Visual and Reverse Extraction attacks, two additional modes of operation have been added to the original InLetSteg algorithm and have been merged into a single one, called as Unified Inter-Letter Steganographic Method, or UILS. The Unified Inter-Letter Steganographic Method (UILS) embeds the data using variable step-size into the host text and the developed mathematical model can calculate the approximate length of the host text required to embed certain data, statistically. In addition, the general mathematical model of UILS makes it customizable to adapt the real-world applications. The statistical parameters that are used through this work are calculated for English host text, but are easily calculable for other languages with similar alphabets and structure of notations. Finally, the programmatically deployed UILS outputs are experimentally examined using 60 attendant and the results are discussed.*

Keywords: *Steganography, UILS, InLetSteg, Reverse extraction attack, Unicode space character, Inter-letter spacing.*

Received June 6, 2018; accepted July 21, 2019
https://doi.org/10.34028/iajit/17/6/1

1. Introduction

The history has a rich record of using steganography as the art of inserting some information in a host medium in a way that the carrier seems unchanged. The word “Steganography” consists of two Greek words: “steganos” which means “protected” and “graph” which means “writing” [18]. Katzenbeisser [8] introduced the first use of steganographic methods to transfer secret information, which dates back to 440 B.C. Most of the ancient steganographic techniques have been applied to text as secret messages but many other carriers such as human shaved head [6], waxed tablets [20], women earrings, texts [17], and images [7, 19, 21] have been used. The aforementioned techniques are mostly applicable in espionage, copyright marketing using watermarking techniques, as well as control of privacy in medical trends [5].

Noting that the Text, as carrier medium, exists everywhere around in both conventional and digital form, it exhibits a good potential to carry the information. As Kessler mentioned, the text-based steganography is divided into two categories: Semagrams and Open Codes [9]. The Semagrams use some symbols to hide information while the open codes hide the secret message by changing the white spaces in building segments of a carrier text. These spaces include inter-word, inter-sentence, inter-paragraph and end-of-line spaces. Semagram was first deployed by Por and Delina [15], but the first published implemented open code steganographic algorithm dates back to 1996 which have been done by Bender *et al*. [2] and his colleagues [8]. Some works suggests manipulating a text as an image and applying image processing beside steganographic techniques. The main advantage of these techniques is their robustness against printing and scanning. Some recent works focus on increasing the amount of covered data [16]. Topkara *et al*. [23] suggested some other techniques for hiding data in a text medium, based on natural language processing [4, 23]. Complexity of encoding and decoding algorithms is the main disadvantage of this method.

This paper aims at proposing a novel steganographic method which uses some basic concepts of some previously proposed methods, i.e., the UniSpaCh, while enhancing their efficiency considerably by defining a new capacity on carrier text. This method is named as Unified Inter-Letter Steganographic Method (UILS). The main merits of UILS include Robustness against “Select All”, “Reverse Extraction” and “DASH” attacks Calculability of approximate required length of text to embed a specific data statistically, Strong mathematical backbone to change the embedding procedure for special application and Customizable data embedding procedure using well-established mathematical backbone, which is a vital trait in especial and real-world applications.

The remainder of this paper is as follows: To shed light on the basic idea behind the UILS method, four major similar and Open Code steganographic methods, namely UniSpaCh [16], SNOW [10], wbStego4open [1, 11] and WhiteSteg [14] are reviewed and their advantages and drawbacks are discussed. In the sequel, the foundations of Inter-Letter steganographic algorithm are presented in section 3. Section 4 discusses how two CSS- and VSS- variant of In LetSteg algorithms are merged into UILS method. Governing formulations are also surveyed too. The implementation method for UILS and the real visual attack experiments procedure are presented in sections 5 and 6. Finally, section 7 concluded the work.

2. Precedent Works

To construct the required logical basis, some important and similar methods to the presented methodology in this work are presented herein. These methods could be considered as logical ancestor of UILS with their inherent advantages and limitations.

"Steganographic Nature of Whitespace" uses the end-of-line spacing to add some information to the carrier text. The payload is added to the carrier by inserting some white space characters to the end-of-line spacing. Each end of line spacing has the capacity of transferring up to 3 bits of payload and the remaining data (if remained) will be added to the end-of-paragraph spacing [10]. This method is originally derived from Bender's Open Code method but offers more capacity to transfer payloads [3, 15]. The original "wbStego4" was first introduced in 1999. This method is one of the developed versions of "wbStego4", which uses inter-sentence and inter-word spacing beside the end-of-line and inter-paragraph spacing to add the payload to the carrier text [1, 15]. The wbStego4open checks the carrier text to ensure that it has sufficient length to cover the payload [11]. This method shows more capacity for transferring payload in compare with SNOW. The efficiency is reported as 1 bit per 8 byte of carrier text in [15].

WhiteSteg is an open code method, which uses inter-word and inter-sentence spacing to add the payload to the carrier. The carrier text will be copied if its length is not enough to cover the message. After implementing this method, Por *et al*. [14] found that it can carry more information in compare with SNOW or the Steganographic Nature of Whitespace, wbStego4open and Spacemimic (not covered in this paper) for equal size of carrier file [13].

Unicode Space Character method, or as abbreviated as UniSpaCh, was first introduced by Por and Delina [15]. This method stands in use of invisible Unicode space characters in inter-word, inter-sentence, inter-paragraph, and end-of-line spacing. The added symbols to the carrier text in SNOW, wbStego4open, WhiteSteg, as well as Spacemimic methods are invisible and the resulted text file is highly robust against visual attack, but the payload can be detected easily using popular and commercial word processor software such as Microsoft Word® and applying DASH Attack, as described in [16]. To realize the suggested algorithm, 8 out of 18 space characters are chosen from Unicode general punctuation chart [24] and two types of data encoding for inter-sentence/inter-word and inter-paragraph/end-of-line spacing are set. The selected Unicode space characters are invisible under DASH arrack. For inter-word and inter-sentence spacing, a set of one or two Unicode space character - that simulates a normal space - can carry 2 bit of payload data. So *n* successive words can carry up to *2(n-1)* bits of payload data. For end-of-line and inter-paragraph spacing, four single Unicode spaces choose to carry 2 bit of data.

3. Inter-Letter Steganographic Algorithm Foundations

To reduce the possible emergence of suspicion by filling the inter-paragraph and end-of-line spacing, the "inter-letter" spacing was defined as a virtual space between two successive letters in a word. These spacing are not real spaces in fact and use of suitable Unicode space characters creates them. The Inter-Letter steganographic, or simply the InLetSteg algorithm, uses inter-letter spacing to embed the payload data. If "Select All" function is applied to the host text, i.e., by applying Ctrl+A in Microsoft Word, any embedded data in inter-paragraph and end-of-line spacing could be detected. Therefore, use of these spacing is possible, but are excluded in this work due to their inherent security limitations.

In applying InLetSteg algorithm, at first, two carrier characters are added to the beginning of host text, respectively. In addition, the end of embedded data is signed by three carrier characters. As the data stream consists of known numbers of 0 and 1, any transition between these two values is presented by a single carrier character. Each of these transitions is called as a Phase Reversal Point (PRP). Therefore, the number of inserted carrier characters is equal to number of PRPs. By examining the various punctuation characters, the "Hair Space" is chosen as carrier character. To this aim, some fonts were examined and the resulted texts were compared with original one. Without knowing the existence of payload in carrier text, detecting the "Hair Space" character are almost impossible. To illustrate the issue graphically, Figure 1 presents the embedding of 40 bits in a short sentence. The carrier character, "Hair space", is shown by vertical grey lines.

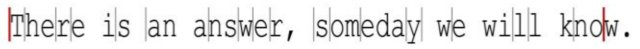
There is an answer, someday we will know.

Figure 1. A sample text with embedded data, the PRPs are shown using thin gray vertical lines.

4. CSS-, VSS- and Unified-InLetSteg

If the cover text consists of *m* characters, ideally it can carry up to *m-1* bits of payload. InLetSteg algorithm searches the beginning of carrier text to insert double "Hair Space" characters. Then the step size, or $S_{insertion}$, is calculated by a defined step size function. The $S_{insertion}$ is defined as the number of characters of the carrier text that carry a bit of embedded data. Finally, the algorithm ends by signing the end of data by a set of three "Hair space" characters. To set the value of step size, there are two different approaches named as Constant Step Size (CSS)-InLetSteg, and Variable Step Size (VSS)-InLetSteg. The UILS, as will be defined later, will merge these two methods together.

The simplest approach to set the value of $S_{insertion}$ is to make it constant. The constant value of step size is defined as follows:

$$S_{inserstion}(k(\varphi), s_c) = s_c \quad ; k(\varphi), s_c \in \aleph \tag{1}$$

The volume of transferred payload by an *m* character word will be reduced to $s_{c-1}(m-1)$ bits. Here the punctuation characters will probably appear in every sc characters if there is PRP. As the "Hair space" character appears rarely in ordinary texts, repeated use of this simple method results in raising suspicion by applying statistical analysis. In addition, steganos-analysis systems can easily guess the used method and extract the embedded data reversely. This method of hacking is called as RE Attack or Reverse Extraction Attack. As the data insertion rate decreases, the resulted text will be more robust against RE Attack, but is definitely vulnerable against it. To reduce the efficacy of RE Attack, the value of $S_{insertion}$ could be varied in respect with an additional reference, such as a set of numbers as key or any valid and repeatable or regenerate-able set of values. The $S_{insertion}$ is defined as:

$$S_{inserstion}(k(\varphi, s_{Max}), s_c) = [k(\varphi, s_{Max})] + s_c \\ ; k(\varphi, s_{Max}), s_c, s_{Max} \in \aleph \tag{2}$$

Where $k(\varphi, s_{Max})$ represents the reference set of numbers and φ is the position of inserted bit. The sc, and sMax is the upper limit of k function. The brackets show the floor function to ensure the output value to be integer. The maximum and minimum value of $S_{insertion}$ will be $s_{Max}+sc$ and sc respectively. The *k* function can be chosen arbitrarily but it must be positive and definite for any value of φ and it is essentially important for $k(\varphi)$ to be, as much as possible, reversely incalculable.

In addition, it is required to check if there is minimum number of zeroes in its first derivative for the selected s_{Max}. This feature ensures that there is minimum number of successive characters with similar value of $S_{insertion}$. Obviously, the $S_{insertion}$ depends on statistical distribution of characters in host text and is variable.

5. Deployment of UILS

The logic of both of methods described in sections 4.1 and 4.2 are identical. Therefore, the CSS- and VSS-InLetSteg methods could be merged into a unified form that eases the software development. The unified version of InLetSteg, the UILS, covers all the properties of consisting versions. To this end, $S_{insertion}$ is proposed as:

$$S_{inserstion}(k(\varphi, s_{Max}), s_c, c_{VSS}) = c_{VSS}[k(\varphi, s_{Max})] + s_c \tag{3}$$

$$k(\varphi, s_{Max}), s_c, s_{Max} \in \aleph;\; c_{Vss} \in \{0,1\} \tag{4}$$

The C_{VSS} activate or deactivate the variability of $S_{insertion}$. When the C_{VSS}=0, the s_c determines the step size of data-insertion algorithm as a constant value. Also while the variable step-size feature is activated (C_{VSS}=1), the value of s_c set a lower limit, equal to or greater than one, to the minimum step size of inserted data, which result in:

$$0 < s_c \le S_{inserstion}(k(\varphi, s_{Max}), s_c, c_{VSS}) \le s_{Max} + s_c \tag{5}$$

As can be seen, the range of changes of $S_{insertion}$ is dependent of s_c and s_{Max} but the distribution of its output value is dependent of *k* function and its properties. The generality of study does not depend on type of *k* function, but is highly dependent on its correct returned values. As long as $S_{insertion}$ generates different values for successive characters, the associated k function is a suitable one. The suitable *k* functions can be categorized as:

- Functions which are in form of $k(\varphi, s_{Max})=u(\varphi);\ 0<u(\varphi)\le s_{Max}$
- Functions which are in form of $k(\varphi, s_{Max})=u(\varphi).s_{Max}+1;\ 0<u(\varphi)\le 1$
- Functions which return the remainder of division as $k(\varphi, s_{Max})=(u(\varphi)$ Mod $s_{Max})+1$

As the alphabetic and numeral characters are placed sequentially in Unicode table, some of the *k* functions, which meet criteria of Equation (4), generate a single value for whole or a part of these characters. So Equation (4) is a necessary but not a sufficient condition. To investigate the issue in its worst case, if φ ranges from 0 to 2^{n-1}, and $s_{Max}=2^{n-1}$ and $s_c=C_{VSS}=1$ the following expression must be met:

$$\left|\frac{\partial S_{inserstion}(k(\varphi, s_{Max}), s_c, c_{VSS})}{\partial k(\varphi, s_{Max})}\right| \ge 1 \tag{6}$$

This ensures that the suggested *k* function returns different values for different successive characters of Unicode table. Obviously various mathematical expressions can be suggested for $k(\varphi, s_{Max})$ without any loss of generality, the $k(\varphi, s_{Max})$ considered to be defined as:

$$k(\varphi, s_{Max}) = \left(C_0 ASC(\varphi)^{p(ASC(\varphi))} \quad Mod \quad s_{Max} \right) + 1 \quad (7)$$

$$p(ASC(\varphi)) = \log_{C_b}^{\left(\frac{C_m C_{UpperLimit}}{ASC(\varphi)}\right)} \quad ; C_{UpperLimit}, C_m, C_b \in \aleph \quad (8)$$

The $ASC(\varphi)$ returns the decimal value of φth character from Unicode table, Mod returns the reminder of division. As shown in Figure 2, the p, as a decaying function, generates smaller returned values for bigger values of $ASC(\varphi)$. If C0, CUpperLimit, C_b and C_m set appropriately, the k function can cover all of the Unicode characters.

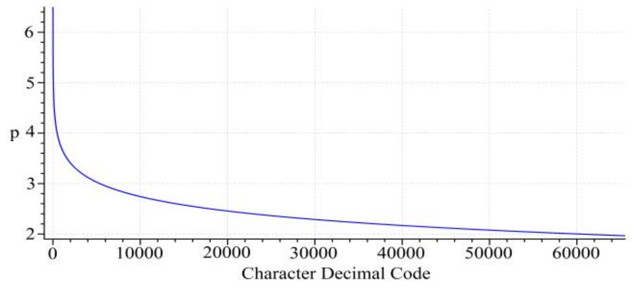

Figure 2. The $p(ASC(\varphi))$ for various UNICODE characters. Its decaying behavior ensures calculable value of k for computational systems with limited resources.

Using (3), (7) and (8) yields in

$$S_{inserstion}(k(\varphi, s_{Max}), s_c, c_{VSS}) = c_{VSS}\left[\left(C_0 ASC(\varphi)^{p(ASC(\varphi))} \quad Mod \quad s_{Max} \right) + 1\right] + s_c \quad (9)$$

Experimental work showed the criteria expressions in Equations (4) and (6) will be meet, as much as possible, if $C_0=0.6$, $C_{UpperLimit}=2^{16}-1$, $Cb=2^{3.49}$ and $Cm=116$. The number of zeroes for first derivative is %0.05 of covered range. This means that less than 35 successive characters out of 2^{16}-1 characters of Unicode table have similar $S_{insertion}$. In addition, the calculation requires 37 bits of memory to handle the probable largest value occurring during the calculation. This is essentially important while one tries to implement the algorithm using very limited amount of resources.

To examine the applicability of the proposed method and to gather the statistical data to analyse the results, UILS algorithm is deployed using C# in Microsoft Visual Studio 2012®. The developed software embeds data in host text using native punctuation characters beside some other characters, which are provided for presentation, and debugging only. As the scientific investigation of UILS requires specific number of bits of data for embedding, the software accepts that in numerical format and generates the requested amount of data randomly. As well, it is possible for the user to upload a file as source of binary data too.

6. Visual Attack Experiment

To investigate the robustness of UILS against visual attack, two experiments are carried out. During the experiments, the attendants were become aware of embedded data in a guided manner. They were questioned about any unusual phenomena or evidence of covered data in host text, and were informed gradually about the position of embedded bits to check if they can observe anything unusual or not. These experiments were Visual Attack against printed host text and Attack Using capabilities of Microsoft Word. The experiment covers 60 men and women, which were familiar with English. Almost non-of them could detect any unusual phenomena, even while using Microsoft Word for DASH and SA attacks.

7. Results and Discussions

To investigate the efficacy and performance of UILS method and to extract essential information for estimation of required length of host text, an important parameter is introduced. The "Character per Bit", namely *CpB*, indicates how many character of host text is employed to transfer a single bit of data. To calculate the *CpB*, some considerations should be noted. As the UILS covers the capabilities of CSS- and VSS-InLetSteg simultaneously, 1KB of randomly generated data is applied to these algorithms. To ensure the generality of study, the number of PRPs are checked to be in vicinity of half of inserted data. The absolute mean value of inserted data has deviation of %0.02441 from nominal mean value, showing the random process which generates the bits is fair enough. The s_c and s_{Max} considered being equal to 2^n as $0 \leq n \leq 11$, $n \in N$. Figure 3 illustrates *CpB* for UILS in constant step size and variable step size modes.

As can be seen, the variable step size capability of UILS increases the efficiency %30 to %50 with respect to the efficiency of CSS-InLetSteg. The slight fluctuation of *CpB* of UILS in Variable Step Size mode is mainly due to the dependency of $S_{insertion}$ on statistical distribution of characters in host text.

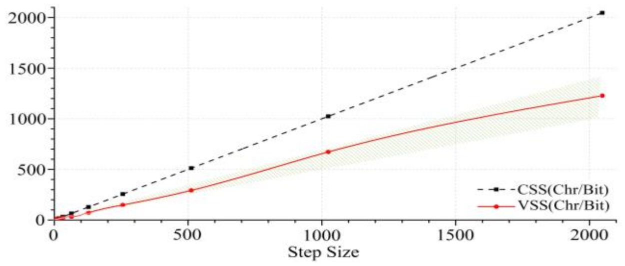

Figure 3. CpB of the UILS in VSS and CSS modes.

The $s_{Max}=2^{11}$ is far beyond the real and applicable requirements and far beyond the capabilities of nominal host texts. Embedding the proposed 1KB of data using $s_{Max}=2^{11}$ will employ 10^7 characters, more than 6000 ordinary book-size pages. Applying CSS with $s_c=2^{11}$ will employ 1.6×10^7 characters. So the curve of CpB for UILS in VSS mode could be considered as linear. To embed b bit of data using UILS in VSS mode with arbitrary s_{Max}, the host text must contain $\varphi_{Max}=0.7bs_{Max}$ characters. The host text must contain $\varphi_{Max}=bs_c$ characters if UILS in CSS mode with arbitrary s_c was used.

To calculate the capacity of UILS method, some statistical calculations are required. To calculate the capacity of a text to embed payload statistically, many types of texts are examined and a value for average number of letters per word is calculated. Table 1 reports these values for different types of texts. Data is extracted using AnyCount® 7.0 software and all source files are converted to Microsoft Word 2010® format. Average number of letters per word, including spaces, ranges from 5.53 to 6.78 letters per word. Suggested value is 6.15 letters per word for English texts and 4.5 letters per word when averaging process preformed over a database, such as reference words of dictionaries [12]. 6.15 letter per word is suitable value for future consideration and is called as "Average Capacity of Word" or C_A. The C_A consists of 5.15 letters per word and a space.

Table 1. The average capacity of words for different type of texts.

Type	Characters Without Spaces	Character With Spaces	Words	Average (Without Space)	Average (With Spaces)
Novel [22]	2632683	3132839	566402	4.64	5.53
Scientific [8]	424313	498063	80324	5.28	6.20
Political [13]	47774	55089	8120	5.88	6.78
Business [19]	6297	7540	1253	5.03	6,01

By applying UILS to a host text, which consists of m words in a paragraph and each word contains C_A letters, the embedded payload can be calculated as

$$D_{UILS} = \left\lceil \frac{(mC_A - 1)}{s_{ave}(s_{Max}, s_c)} \right\rceil \text{(Bits)} \tag{10}$$

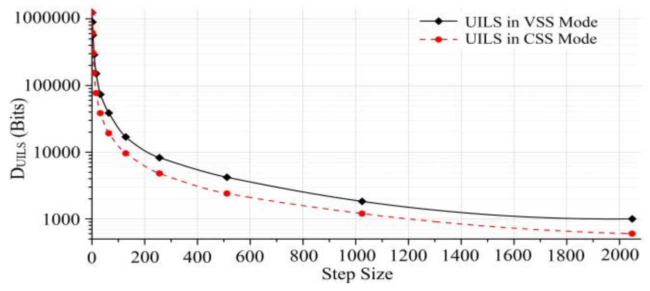

Figure 4. The capacity of UILS method in VSS and CSS modes to embed data in a text that consist of 200,000 words.

The s_{ave} is the statistical average value of step size for specific s_{Max} and s_c. The actual value of S_{ave} can be calculated statistically using appropriate values of s_{Max} and s_c to embed large amount of data, in host text which is long enough. Figure 4 illustrates the capacity of a long text to embed data.

8. Conclusions

A novel steganographic method was introduced to embed binary data into text. Noting that CSS and VSS InLetSteg methods are two suitable algorithms, the proposed method merged them into UILS algorithm with enhanced functionality with respect to the previous ones. The statistical investigation of the proposed algorithm revealed that UILS in VSS mode is %30 to %50 more efficient than when is in CSS mode with comparable step-size. In addition, the theoretical capacity of UILS method was justified. Excluding the End-of-Line and Inter-paragraph spacing to carry the embedded data, capability to activate VSS- and CSS- modes and finally the inability of standard word processors to mark the selected carrier characters have made the UILS highly robust against SA attack, RE attack, and DASH attack. In addition, the experimental investigation of the UILS method, using 60 attendants, revealed its robustness against Visual attack and approved the selection of the selected carrier character. The complexity of the algorithm is highly dependent on optimistic selection of $S_{insertion}$. The selected procedure in Equation (9) is one of the possible choices out of many and has selected to show the capability of method and to extract some basic features of it, as described in Equations (5) and (6). This is why the brute force has not been calculated.

References

[1] Bailer W., "WbStego Steganography Tool" [web site], http://wbs- tego.wbailer.com, wbStego4open overview, Last Visited, 2018.

[2] Bender W., Gruhl D., Morimoto N., and Lu A., "Techniques for Data Hiding," *IBM Systems Journal*, vol. 35, pp. 313-336, 1996.

[3] Culnane C., Treharne H., and Ho A., "A New Multi-Set Modulation Technique for Increasing Hiding Capacity of Binary Watermark for Print and Scan Processes," *in Proceedings of the 5th International Conference on Digital Watermarking*, Germany, pp. 96-110, 2006.

[4] Hamad N., "Hiding Text Information in a Digital Image Based on Entropy Function," *The International Arab Journal of Information Technology*, vol. 7, no. 2, pp. 146-151, 2010.

[5] Hernandez-Castro J., Blasco-Lopez I., Estevez-Tapiador J., and Ribagorda-Garnacho A., "Steganography in Games: A General Methodology and its Application to the Game of Go," *Computers and Security*, vol. 25, no. 1, pp. 64-71, 2006.

[6] Herodotus, *the Histories*, J. M. Dent and Sons Ltd., 1992.

[7] Hossain M., Al Haque S., and Sharmin F., "Variable Rate Steganography in Gray Scale Digital Images Using Neighborhood Pixel Information," *The International Arab Journal of Information Technology*, vol. 7, no. 1, pp. 34-38, 2010.

[8] Katzenbeisser S., *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House, 1999.

[9] Kessler G., "An Overview of Steganography for the Computer Forensics Examiner," *Forensic Science Communications*, vol. 6, no. 3, pp. 1-27, 2004.

[10] Kwan M., "The SNOW Home Page," http://www.darksi- de.com.au/snow, Last Visited, 2018.

[11] Murphy B., "Syntactic Information Hiding in Plain Text," Master Thesis, Trinity College, 2011.

[12] Pierce J., "An Introduction to Information Theory: Symbols, Signals and Noise," *Dover Publications*, pp. 75, 1961.

[13] *Political Declaration on HIV and AIDS: Intensifying Our Efforts to Eliminate HIV and AIDS*, United Nations, A/RES/65/277, Resolution adopted by the General Assembly, USA, 2011.

[14] Por L., Ang T., and Yin D., "WhiteSteg: A New Scheme in Information Hiding Using Text Steganography," *WSEAS Transactions on Computers*, vol. 7, no. 6, pp. 735-745, 2008.

[15] Por L. and Delina B., "Information Hiding: A New Approach in Text Steganography," *in Proceedings of the 7th WSEAS International Conference on Applied Computer and Applied Computational Science*, Hangzhou, pp. 689-695, 2008.

[16] Por L., Wong K., and Chee K., "UniSpaCh: A Text-Based Data Hiding Method Using Unicode Space Characters," *The Journal of Systems and Software*, vol. 85, no. 5, pp. 1075-1082, 2012.

[17] Rahman M., Khalil I., Yi X., and Dong H., "Highly Imperceptible and Reversible Text Steganography Using Invisible Character based Codeword," *in Proceedings of the Pacific Asia Conference on Information Systems, PACIS Proceedings, 21st Pacific Asia Conference on Information Systems*, Langkawi, pp. 1-13, 2017.

[18] Reddy H. and Raja K., "Wavelet based Non LSB Steganography," *International Journal of Advanced Networking and Applications*, vol. 03, no. 3, pp. 1203-1209, 2011.

[19] Schuman M., "Spain's Death Spiral and the Hypocrisy of the Euro," *Times Business*, vol. 179, no. 15, 2012.

[20] Tacticus A., *Aineias the Tactician: How to Survive under Siege*, Clarendon Press, 1990.

[21] Tasheva A., Tasheva Z., and Nakov P., "Image Based Steganography Using Modified LSB Insertion Method with Contrast Stretching," *in Proceedings of the 18th International Conference on Computer Systems and Technologies*, New York, pp. 233-240, 2017.

[22] Tolstoy L., *War and Peace*, *A Penn State Electronic Classics Series Publication*, Pennsylvania State University, 1805.

[23] Topkara M., Topkara U., and Atallah M., "Words are Not Enough: Sentence Level Natural Language Watermarking," *in Proceedings of ACM Workshop on Content Protection and Security*, Santa Barbara, pp. 37-46, 2006.

[24] Unicode Standard version 6.1, General Punctuation, Supplement Punctuation, Range 2E00-2E7F, *Unicode Inc. Press*, USA, 2011.

Ahmad Esmaeilkhah was born in June, 1981. He received his B.Sc. in 2007, M.Sc. in 2013 in electrical engineering and his Ph.D. in 2019 in Telecommunication Engineering. He is now with University of Urmia. The main areas of his interest are Antenna, Scaling of Electromagnetic Structures, Radar and Steganography.

Changiz Ghobadi was born in June, 1960 in Iran. He received his B.Sc. in Electrical Engineering Electronics and M.Sc. degrees in Electrical Engineering Telecommunication from Isfahan University of Technology, Isfahan, Iran and Ph.D. degree in Electrical-Telecommunication from University of Bath, Bath, UK in 1998. Now he is a Professor in the Department of Electrical Engineering of Urmia University, Urmia, Iran.

Javad Nourinia received his B.Sc. in Electrical and Electronic Engineering from Shiraz University, M.Sc. degree in Electrical and Telecommunication Engineering from Iran University of Science and Technology, and Ph.D. degree in Electrical and Telecommunication from University of Science and Technology, Tehran Iran in 2000. Now he is a Professor in the Department of Electrical Engineering of Urmia University, Urmia, Iran.

Maryam Majidzadeh was born in 1987 in Urmia, Iran. She received her B.Sc., in electrical engineering from Urmia University in 2009. As well, she received her M.Sc. and Ph.D. degrees in communication engineering from the same university in 2012, and 2016 respectively. She is now assistant professor in Department of Electrical and Computer Engineering, Urmia Girls Faculty, West Azerbaijan branch, Technical and Vocational University (TVU), Urmia, Iran.