JOURNAL of COMPUTER SCIENCE

A Novel Text Steganographic Technique Using Specific Alphabets

May Htet[1*] and Su Wai Phyo[2]

[1]*Ph.D Student/Department of Computer Engineering and Information Technology/Mandalay Technological University, Mandalay, 100/107, Myanmar*

[2]*Faculty/ Department of Computer Engineering and Information Technology/Mandalay Technological University, Mandalay, 100/107, Myanmar*

Email: [1]mayhtet@iuj.ac.jp, [2]suwaiphyo@gmail.com

Abstract

In today's electronic era, wealth of electronic information are being accessed over the Internet. Several important information and private data transferring over the Internet are being hacked by attackers via latest communication technologies and as such, maintaining the security of secret data has been a great challenge. To tackle the security issue, cryptographic methods as well as steganographic techniques are essential. This paper focuses on hybrid security system using cryptographic algorithm and text steganographic technique to achieve a more robust security system. In this work, to overcome the limited data hiding capacity, suspiciousness, and data damaging effect due to modification of traditional steganographic techniques, a new technique for information hiding in text file is proposed. The proposed approach conceals a message, without degrading cover, by using the first, second, second last, and last letter of words of the cover text. Hence, from the embedding capacity point of view, its capacity depends on the similarity of characters of the words in cover text. In addition, as a further improvement for security, secret message encryption is performed using the Blowfish algorithm before hiding into the innocuous cover text.

Keywords: Blowfish Algorithm, Cryptography, New embedding technique, Steganography, Text steganography.

1. INTRODUCTION

At present, with the advances in the Internet and communication technology, security of multimedia data and confidential digital resources transferring over the Internet is becoming an important matter. As a result, various security techniques have been applied to provide some security requirements such as confidentiality, integrity, authentication, and non-repudiation and to ensure the secure information exchange between the different parties. Cryptography and steganography are key techniques used to secure data transfer over the Internet.

Cryptography is a method of transferring data from a dispatcher to a target receiver by altering it into an incomprehensible form using a secret code [1]. It is the practice of protecting only the contents of a message and nobody can interpret its content. Thus, when the secret message is sent, its existence can be visible by anyone. Although how the enciphered message is prepared indestructible, the perceptibility of its existence can still attract the consciousness of potential attackers. On the other hand, steganography deals with making secret data invisible by hiding it in

different cover media, hosts or carriers [2]. It is concerned with concealing the fact that a secret message is being sent. Therefore, the opponents have no idea about hiding the covert message inside carriers. The advantage of steganography is that it gives permission to send messages without making suspicion. However, the message is readable if a rival party discovers it. Therefore, to address these issues of cryptography and steganography, these two techniques are integrated together as a complement of each other. By integrating cryptographic algorithm with stegnographic technique, it is possible to protect the contents of a message as well as its existence.

Cryptographic techniques can be mainly classified into symmetric or private key algorithms and asymmetric or public key algorithms. With symmetric algorithms, both the sender and receiver use his or her private key for data sending and receiving. In public key cryptography, encryption and decryption use different keys, a public key and a private key. In this attempt, a symmetric cryptographic algorithm, Blowfish is used for ciphering the secret data.

At the steganographic point of view, the steganographic algorithms employ text, image, audio or video files as the medium to ensure hidden exchange of information between multiple contenders and to protect the data from the eavesdroppers. In this paper, only a text file is considered as cover medium and a new efficient embedding method is employed for hiding the encrypted message into the cover text in order to get the double layer of security for secret message.

The rest of the paper is organized as follows: Section 2 depicts the general model of proposed system. Section 3 mentions the general structure and operation of Blowfish algorithm. Section 4 explains the proposed embedding technique. Section 5 describes the implementation results of the proposed system. Section 6 shows the performance analysis of the proposed approach. Section 7 discusses the merits and demerits of the proposed approach. Section 8 draws the conclusion.

2. THE PROPOSED SYSTEM MODEL

Figure 1 shows the proposed system model. The model consists of two portions: sender side and receiver side.

At the sender side, the secret message is ciphered using the Blowfish encryption algorithm. Then the cipher text is concealed in the cover text using the proposed embedding algorithm and generates the stego text and stego key. The stego text and stego key are sent to the receiver over the public network.

At the receiver, the stego text together with the stego key from the sender is used to extract the hidden cipher text from the stego text. After that, the Blowfish decryption algorithm is used to retrieve the original secret message from cipher text.

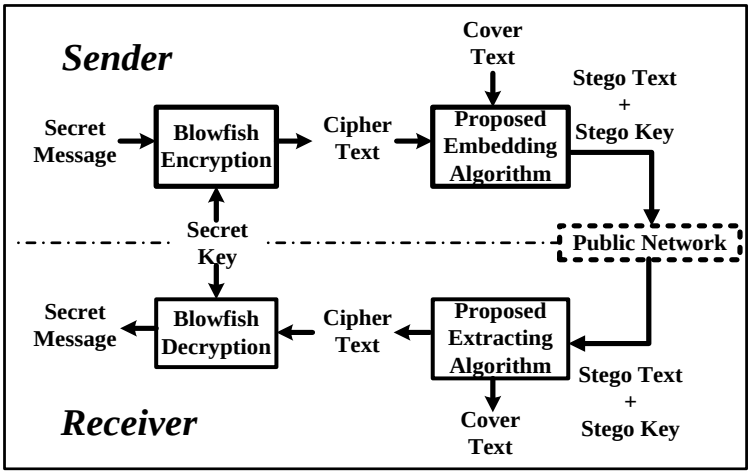


FIGURE 1: Proposed system model

3. BLOWFISH ALGORITHM

Blowfish is a symmetric block cipher, designed by Bruce Schneier in 1993 and used as a replacement for DES or IDEA algorithm. Blowfish has a 64-bit block size and a variable key length from 32 up to 448 bits. This algorithm is suitable for applications in which the key does not change very frequently. The default key size is 128 bits. Variants of 16 round or less are available in Blowfish. Distinguished features of the design in Blowfish include key-dependent S-boxes and a highly complex key schedule.

According to recent research in [3], Blowfish outperforms some existing symmetric algorithms (DES, 3DES, RC2, RC6, AES) in terms of processing time, throughput, and power consumption. The algorithm consists of two parts: a key expansion part and data encryption part.

3.1 Key expansion part

Blowfish uses a large number of subkeys. These keys must be pre-computed before data encryption and decryption. The algorithm keeps two sub-key arrays: the 18 (32-bit) P-array and four 256 (32-bit) Substitution boxes (S-boxes). The P-array and S-boxes are first initialized with a random number, which was the hexadecimal fraction of pi. Then the P-array values are modified by the key using an XOR operation. The P-array and S-boxes are then modified by using the encryption algorithm. After the initialization, 64-bits plaintext data can be encrypted using the 16 round Feistel network. Key expansion converts a key of at most 448 bits into several subkey arrays totaling 4168 bytes.

3.2 Data encryption part

Figure 2 depicts the encryption operation of Blowfish algorithm [4]. The input plaintext is divided into two 32-bit halves LE0 and RE0. The encryption algorithm can be defined by the following pseudo code.

For i = 1 to 16 do

$LE_i = F[LE_{i-1} \oplus P_i] \oplus RE_{i-1}$;

$RE_i = LE_{i-1} \oplus P_i$;

$LE_{17} = RE_{16} \oplus P_{18}$;

$RE_{17} = LE_{16} \oplus P_{17}$;

The resulting ciphertext is contained in the two variables LE17 and RE17.

Decryption process is exactly the same as encryption process, except that P entries, P1, P2, ..., P18 are used in the reverse order.

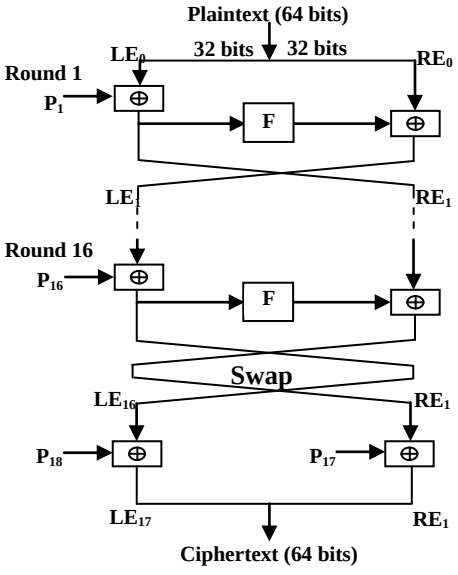


FIGURE 2: Block diagram of Blowfish encryption algorithm

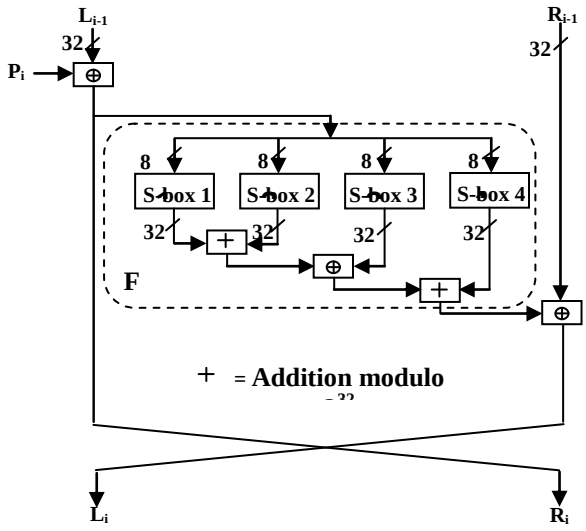


FIGURE 3: F function of Blowfish algorithm

In Figure 3, the function F splits the 32 bit input into four 8-bit quarters: a, b, c, and d and uses the quarters as input to the S-boxes.

$$F(a, b, c, d) = ((((S1, a + S2, b) \bmod 2^{32}) \oplus S3, c) + S4, d) \bmod 2^{32} \quad (1)$$

Thus, each round includes the complex use of addition modulo 2^{32} and XOR, plus substitution using S-boxes.

4. THE PROPOSED EMBEDDING TECHNIQUE

The approach makes use of a pre-determined cover text which can be any meaningful piece of English text and can be taken from any source (e.g. a paragraph from a newspaper/magazine/book). It works on the binary value of a character and performs message hiding using first (f), second (s), second last (s′), and last (l) letters of the words of a cover text.

Firstly, the secret message must be converted into its binary equivalent. Then, each bit is read in order from the binary file. After that, each word is picked up sequentially from the cover text and it is written down in the stego text. Then, each word will be checked for two times.

The first check is to make sure that the first and last letter of the word of the cover is different. The word with same first and last letter is not considered for hiding the secret bit. If the secret bit is 0, then write the first letter of the word in the stego key. Otherwise, write the last letter of the word in the stego key.

The second check is to confirm that the second and second last letter of the word of the cover text is different. Like the first check, a word having the same second and the second last letter will be skipped for concealing the consecutive bit. Write the second letter in the stego key if the secret bit is 0 and or else, write the second last letter in the stego key.

The message hiding process is performed till the ending of the binary file. If there no more words are left left in the cover text during hiding process, the word must be read again from the start of cover text. Nevertheless, the words will not be written again and again in the stego text for the subsequent times while writing only the corresponding characters either first, second, second last or last letter in the stego key depending on the bit to be concealed.

As a result, the indistinguishable stego text in appearance as the cover text will be produced at the end of the embedding process as there is no change or modification to the cover text in hiding process.

In extracting process, we first get the stego text and the stego key. Then, each character is read the from stego key. Next, we pick a word in order from the stego text and it is written down in the cover text. Then, each word is checked for two times. After that, the reverse procedure is performed by comparing the words in the stego text with the characters in the stego key. The extracting process is repeated until the end of stego key. If there are no more words in the stego text to compare with the character in the stego key, the word is read from the beginning of the stego text, repetitively.

Algorithm for message embedding is as follows.

1. Get a cover file and set count = 0.
2. Convert the letters in input message file to its binary equivalent (bin).
3. Read a bit (x) from the bin.
4. Read a word from the cover file (If the size of cover file is not enough for hiding the secret message, read the word from the beginning of the cover file by doing count+=1) and if count = 0, write it in the stego file. (f = first letter, l = last letter, s = second letter, and s´= second last letter of the word.)

5. If first and last letter of the word is the same, then go to step 4.
6. If x = 0, write “f” in the stego key.
7. Else if x = 1, write “l” in the stego key.
8. If second and second last letter of the word is the same, then go to step 3.
9. Read a bit (x) from the bin.
10. If x = 0, write “s” in the stego key.
11. Else if x = 1, write “s´” in the stego key.
12. Repeat steps 3 to 11 till the end of the bin file.
13. Send the stego file and the stego key to the receiver.

Algorithm for message extraction is as follows.

1. Get the stego file and stego key and set count = 0.
2. Read a character (c) from the stego key.
3. Read a word from the stego file (If there is no more word in the stego file to compare with the characters in the stego key, then read the word from the beginning of the stego file by doing count+=1.) and if count = 0, write it in the cover file. (f = first letter, l = last letter, s = second letter, and s´= second last letter of the word.)
4. If first and last letter of the word is the same, then skip that word and go to step 3.
5. If c = f, then bit b = 0.
6. Else if c = l, then bit b = 1.
7. Write b in a file.
8. If second and second last letter of the word is the same, then go to step 2.
9. Read a character (c) from the stego key.
10. If c = s, then bit b = 0.
11. Else if c = s´, then bit b = 1.
12. Write b in a file.
13. Repeat steps 2 to 12 till the end of the stego key.
14. Convert the file into its character equivalent.
15. Obtain the original message.

As an example to present this steganography process in detail, consider a secret message “Hello World!” to be hidden in a cover file. After enciphering the message, the cipher text generated was “ěxśïv!KÀ¶ĊtP”. Before hiding into the cover text, the cipher text is converted into its binary equivalent.

Figure 4 and 5 show the example of data hiding and seeking processes of the proposed system.

Figure 6 shows the selected cover text in which the secret binary file is to be hidden. Figure 7 shows the stego text after hiding the binary file. It can be seen that the appearances of two text files are exactly the same.

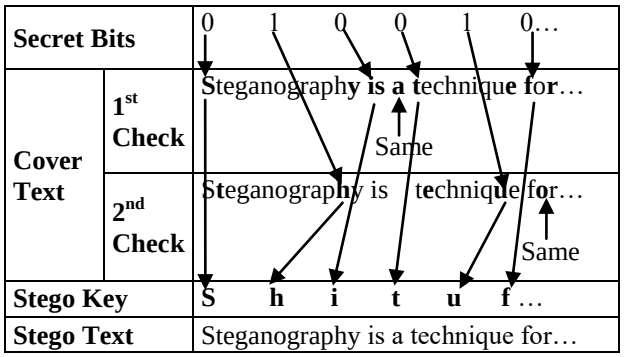


FIGURE 4: Data embedding process

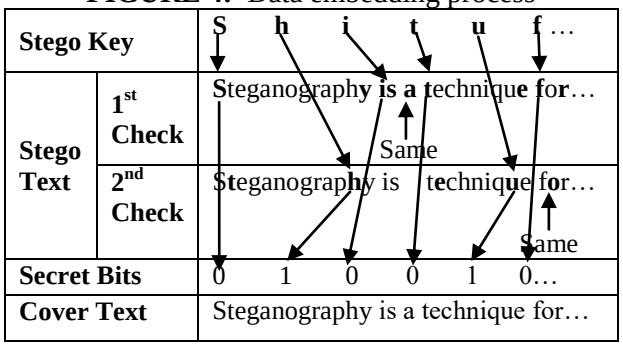


FIGURE 5: Data extraction process

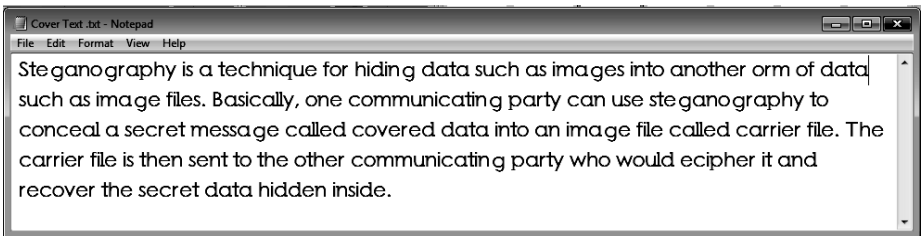


FIGURE 6: Cover text

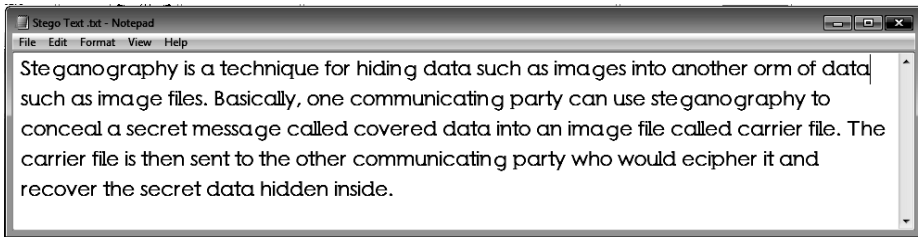


FIGURE 7: Stego text

5. SYSTEM IMPLEMENTATION

This section describes the implementation results with a series of interface. The main interface of the proposed scheme is illustrated in Figure 8.

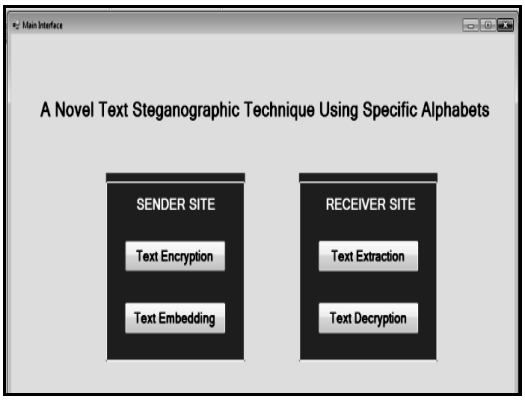


FIGURE 8: Main interface

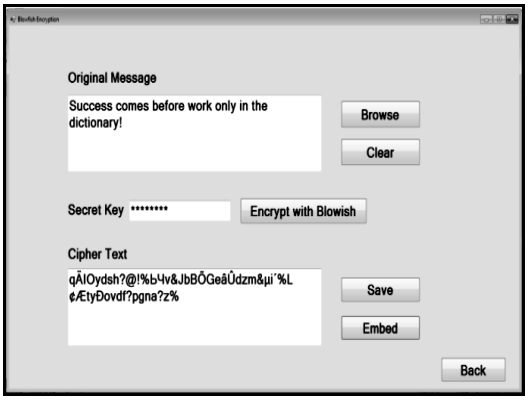
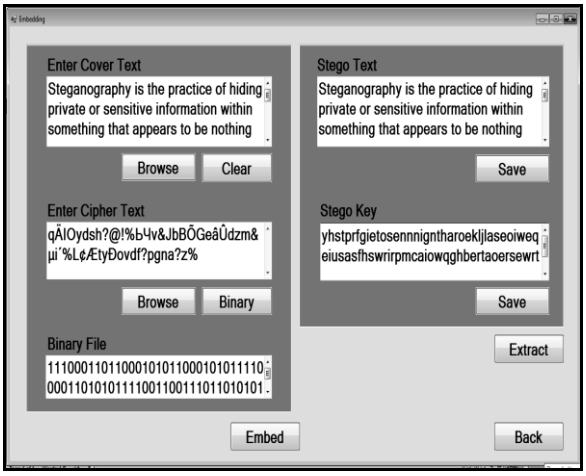


FIGURE 9: Text encryption

At the sender site, there are two functions: text encryption and embedding. The encryption process is shown in Figure 9. Firstly, the user loads the plain message and creates it into cipher text using Blowfish algorithm with the help of secret key. After that, the user has to click the "Embed" button to perform text embedding as shown in Figure 10.



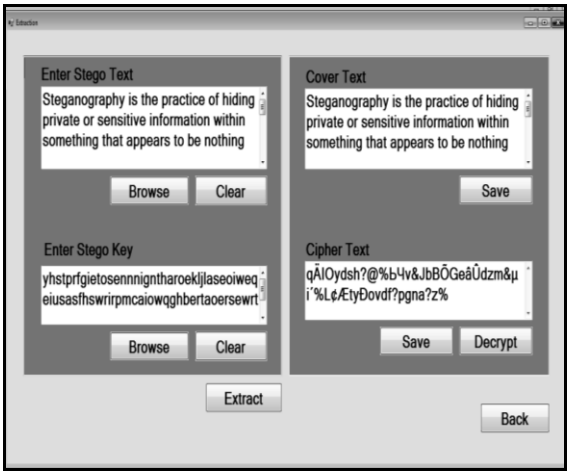


FIGURE 10: Text embedding **FIGURE 11:** Text extraction

In embedding process, the user has to load the cover text and cipher text files. Before hiding into the cover, the cipher text must be converted into binary form by clicking the "Binary" button. Then, with the help of the "Embed" button, the binary file is embedded into the cover text and produces the stego text together with the stego key. After that, by pressing the "Extract" button, text extracting process can be performed as demonstrated in Figure 11.

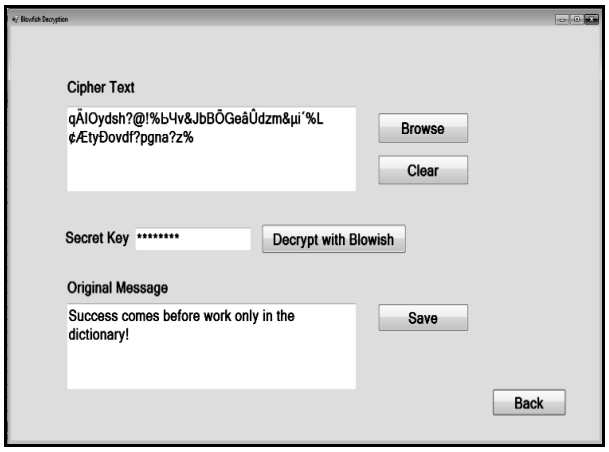


FIGURE 12: Text decryption

At the receiver site, extracting process and decryption process are performed by using the reverse orders of the sender's processes to obtain the original plain text message as shown in Figure 11 and 12.

6. PERFORMANCE ANALYSIS

To measure the effectiveness of the proposed embedding algorithm, it is considered with three aspects: capacity consideration, similarity measures of cover text and stego text, and security consideration.

6.1 Capacity consideration

Capacity is the amount of secret information that is able to be hidden in the cover text. The capacity ratio is computed as the amount of hidden bytes divided by the size of the cover text in bytes. Capacity ratio is usually represented by percentage [5].

$$\text{Capacity ratio} = \text{Amount of hidden bytes} / \text{Size of the cover text in bytes} \quad (2)$$

Generally, the proposed method can hide two bits of secret message using one word. In addition, the words are used repetitively from the beginning of the cover file if its size is not enough. Hence, using the proposed approach, any amount of secret message can be hidden using the fixed size of cover text and it can be assumed that the larger the size of secret message, the greater the percentage capacity of the proposed approach. Therefore, the proposed approach satisfies hiding capacity requirement.

6.2 Similarity measures of cover text and stego text

Similarity metrics are used to measure distance between two values or sequence. As the aim of steganography is to hide the existence of secret data, similarity between cover text and stego text is important to show that the cover file hasn't been modified due to embedding. In this paper, two methods are applied for comparing the similarity between cover text and the stego text:

- Jaro-Winkler similarity metric
- Histogram analysis

6.2.1 Jaro-Winkler similarity metric

The Jaro-Winkler metric (Jaro score), which is a variant of the Jaro distance is a measure of similarity between two strings. The Jaro-Winkler distance metric is designed and best suited for short strings such as person names. It is mainly used in the area of record linkage (duplicate detection). The higher the Jaro score for two strings is, the more similar the strings are. The score is normalized such that 0 equates to no similarity and 1 is an exact match [6]. The Jaro-Winkler metric is calculated as:

$$\text{Jaro-Winkler}(s1,s2) = \text{Jaro}(s1,s2) + (L * p\,(1 - \text{Jaro}(s1,s2))) \quad (3)$$

$$\text{Jaro}(s1,\ s2) = \frac{1}{3} * \left(\left\lfloor \frac{m}{|s1|} \right\rfloor + \left\lfloor \frac{m}{|s2|} \right\rfloor + \frac{m-t}{m}\right) \quad (4)$$

, where s1 and s2 are the two strings whose similarity is to be measured, L is the length of common prefix (maximum 4 characters), p is scaling factor whose standard value is 0.1, m is the number of matching characters and t is the number of transpositions. Two characters from s1 and s2 respectively are considered matching only if they are not farther than $\left\lfloor \frac{\max(|s1|,|s2|)}{2} \right\rfloor - 1$. Each character of *s*1 is compared with all its matching characters in s2. The number of matching (but different sequence order) characters divided by two defines the number of transpositions.

In the proposed embedding approach, since data is hidden without altering the cover text, the cover and its corresponding stego text are exactly the same in appearance leading to the Jaro score of "1".

6.2.2 Histogram analysis

Histograms can be used to measure the effectiveness of the data hiding algorithm. It can be assumed that the embedding algorithm is efficient if the histograms of the two text files are still similar after the embedding process [7]. In this work, it has been examined that the histograms of

the cover text and the stego text are identical in appearance as no change is made to the cover text in hiding the data. Therefore, it can be said that the proposed embedding algorithm is efficient. Histograms of cover text and stego text are depicted in Fig. 13 and 14.

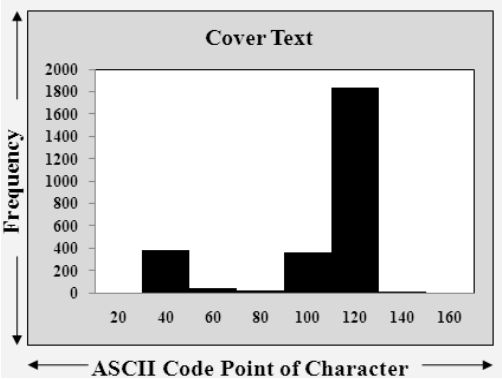


FIGURE 13: Histogram of Cover Text.

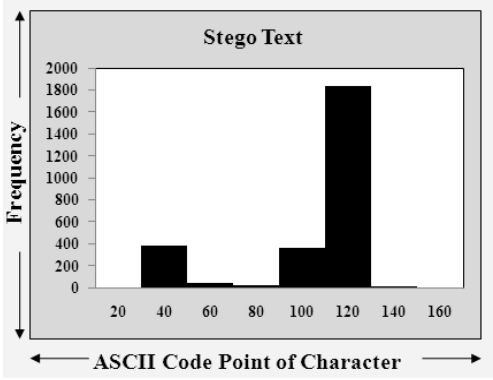


FIGURE 14: Histogram of Stego Text.

6.3 Security consideration

The proposed method produces the same stego text in appearance as the cover text. Thus, it will not reveal to public about the existence of any hidden data. Even if a snooper obtains both cover and stego files, any difference between the two files will not be figured out. Hence, it satisfies security aspect of the steganography. In addition, in this proposed approach, four specific letters are used for hiding the secret bit. Therefore, at the security point of view, it is impossible for anyone to extract the hidden data from the stego media without knowing the stego key. Besides, the use of pre-encoding method before embedding operation enhances the security level of the secret message. For this reason, even though the third party detects the existence of the secret communication, the requirement of cryptographic key causes the complicated retrieval process of original message.

7. DISCUSSION

In any steganographic system, there are three main performance parameters: capacity, security and robustness. Capacity deals with the amount of data that can be hidden in a cover, compared to the size of the cover [8]. Security refers to the ability of making the hidden information undetectable by an interceptor [9]. Robustness refers to the ability to prevent the detection of concealed data in the cover text and the removal of data due to modification [10].

In this proposed approach, a large cover file is not required to be considered even if the larger secret message is hidden. Therefore, its data hiding capacity would be higher than most of the existing approaches which work on the binary value of embedded characters. However, it is time consuming due to the repetitive use of words from the cover text when the cover size is not enough for hiding the secret data. On the other hand, the embedding capacity of the proposed approach depends on the similarity of characters of the words in cover text in the case of without looping the cover text.

The proposed approach performs message hiding in any natural looking meaningful piece of English Text, thus the stego text will not raise suspicion and this property makes it difficult for steganalysis. In addition, as an additional improvement of security level, the secret message is first scrambled using Blowfish algorithm and the resulting cipher text is then hidden in a cover. Although, the same secret key is used in both encryption and decryption processes, if different secret key is used for enciphering the same message, the different cipher text will be obtained. So, without knowing the exact secret key, no one can reveal the original secret message from the cipher text.

The proposed approach does not insert additional spaces or uses misspelled words and extended letter points in embedding process. Even if the modification to the stego text is performed by using character recognition technique, it can resist to the loss of hidden data. If the system is not secure and robust, larger embedding capacity will be useless. Therefore, the proposed approach fulfils all the three main features of steganography.

8. CONSLUSION & FUTURE WORK

A new embedding algorithm for information hiding in text is presented in this paper. The proposed algorithm does not make changes or modifications to the format or characteristics of the cover text to hide secret data in it. Embedding is done by using specific letters of a word and writing the corresponding characters directly in the stego key according to the bit sequence to be secreted. As a result, the generated stego text contains natural looking information and it avoids suspicion regarding the hidden data inside it. Besides, unlike other existing approaches, the proposed algorithm has no data damaging effect due to modification such as retyping or opening the stego text with a word processor program. Moreover, in the proposed algorithm, even if larger secret message is hidden, there is no need to consider the larger cover size as it reads the word from the beginning of the cover text repetitively if the size of cover text is not enough for hiding the large amount of data. Therefore, the larger size of secret message leads to the higher data hiding capacity of the proposed approach.

As this method features security, capacity, and robustness, the three needed aspects of steganography, it is useful for setting up a reliable communication link between different partners and exchanging information secretly using text documents. The proposed system can be applied to transmit private digital data such as secret chemical formula of a food industry or secret plan of a new mission of a military, etc. safely over the public communication channel. The proposed system can only access text data format (.txt). As further extensions, the concept of proposed approach can be applied in other languages with little modification. In addition, the concept of proposed approach can be extended to develop new embedding techniques for more efficient data hiding and for any media cover format such as image, audio, and video.

9. REFERENCES

[1] Dorothy Elizabeth Rob et.al, Cryptography and data security, *International Journal of Engineering Technology*, vol. 1, no. 2, pp. 106-113, 2011.

[2] A. A. Gutub and M. M. Fattani, A novel Arabic text steganography method using letter points and extensions, *International Journal of Computer, Information, Systems and Control Engineering*, vol. 1, no. 3, pp. 483-486, 2007.

[3] M. Mathur and A. Kesarwani, Comparison between DES, 3DES, RC2, RC6, Blowfish, and AES, *National Conference on New Horizons in* IT (NCNHIT), 2013.

[4] M. A. Kumar and S. Karthikeyan, Investigating the efficiency of Blowfish and Rejindael (AES) algorithms, *International Journal of Computer Network and Information Security*, vol. 4, no. 2, pp. 22-28**,** 2012.

[5] F. A. Haidari, A. Gutub, K. A. Kahsah and J. Hamodi, Improving security and capacity for Arabic text steganography using "Kashida" extensions, *International* Conference on Computer Systems and Applications, pp. 396-399, 2009.

[6] S. D. Pandya and P. V. Virparia, Testing various similarity metrics and their permutations with clustering approach in context free data cleaning, *International Journal of Computer Science and Security*, vol. 3, pp. 344-350, 2009.

[7] S. Saleh, A secure data communication system using cryptography and steganography, *International Journal of Computer Networks & Communications*, vol. 5, no. 3, May 2013.

[8] K. A. Kumar, S. Pabboju and N. M. S. Desai, Advance text steganography algorithms: an overview, *International Journal of Research and Applications*, vol. 1, no. 1, pp. 31-35, 2014.

[9] H. Singh, P. K. Singh and K. Saroha, A survey on text based steganography, *Proceedings of the 3rd National Conference on Computing for Nation Development*, India, 2009.

[10] L. Y. POR and B. Delina, Information hiding: a new approach in text steganography, 7th WSEAS International Conference on Applied Computer & Applied Computational Science (ACACOS08), Hangzhou, China, 2008.

May Htet was born in Myanmar, in 1984. She received the B.E degree in Information Technology from the Technological University (Hmawbi), Yangon Division, Myanmar in 2006, and MBA degree in ICT concentration from the International University of Japan (IUJ), Minamiuonuma city, Niigata prefecture, Japan in 2012. She is currently pursuing the Ph.D. degree in Computer Engineering and Information Technology at the Mandalay Technological University (MTU), Mandalay Division, Myanmar. From 2007 to 2012, she was an Assistant Lecturer in Technological University (Mawlamyaing), Mon State, Myanmar. Presently, she is a Lecturer in computer science at the Mandalay Technological University (MTU). Her research interest includes Information Security, Network Management, and Data Mining.

Su Wai Phyo was born in Myanmar, in 1980. She received the B.E degree in Information Technology from the Mandalay Technological University (MTU), Mandalay Division, Myanmar in 2002. She got her M.E degree with IT specialization from Yangon Technological University (YTU), Yangon Division, Myanmar in 2004. She obtained her Ph.D degree from Mandalay Technological University (MTU) in 2008. Currently, she is working as an associate professor for Department of Computer Engineering and Information Technology, Mandalay Technological University (MTU). She has over fifteen years working experience in teaching. Her research interest includes Information Security and Distributed System.