Computer Science and Engineering 2012, 2(5): 46-57
DOI: 10.5923/j.computer.20120205.02

A Language Independent Platform for High Secured Communication Using Encrypted Steganography

Vijayaratnam Ganeshkumar[1,*], Ravindra L. W. Koggalage[2]

[1]Creative Technology Solutions Pte Ltd, Sri Lanka
[2]Deputy Vice Chancellor (Academic), General Sir John KotelawalaDefence University, Sri Lanka

Abstract Computer usage is increasing, for both social and business areas, and it will continue to do so. This naturally leads to an increase in the way in which we as individuals and organizations we work for may be attacked.Increase of cyber-crime has compelledto re-think for a secure communication medium in our day-to-day life. Protecting the information during transmission is a foremostchallenge against eavesdroppers. Encryption is one of the widely used techniques that ensure the security of the message. However, sending encrypted messages often draw eavesdropper's attention. Steganography is a method of letteringsecret messages in a way that nobody except for the sender and the recipient would suspect the existence of the hidden message. In other words, it's an art/science of hiding messages. Steganography is often combined with cryptography so that even if the message is discovered it cannot be read.Historical steganography involved techniques such as disappearing ink or microdots. Modern steganography involves in computer files such as images, audio, video files and even in text documents. However, each stereographic technique is focused in just one medium; such as image or audio file and etc. And each steganographic algorithms had independent implementation; as a resultthere is no generic framework or application that will produce stegano medium for an end-user. In this paper, a generic framework is presented, which provides generic steganographic functionality via well-defined Application Programming Interface (API). So that the advanced developers can develop framework modules, this can be used by the end-users directly as stegangraphicapplication. And also this paper describes a newly invented steganographic technique which utilizes the inter-character spacing of a Rich-Text-Format (RTF) document. This technique can be used to transmit concealed messages in multi-language or combination of languages, which can be represented in Unicode. As an additional optimization technique; user defined code based (UDC) technique is also proposed to achieve compression of Unicode languages. Another advantage is that this technique can be used to send multi different language secret messages through the communication channel.

Keywords Steganography, Cryptography, Security, Unicode, Multi-language, Encryption, Eavesdropper, Framework

1. Introduction

Over the last few years, the usage of internet has been dramatically increased from youngsters to an expert. Not only internet has become an essential part of our daily life, but also been a reason for increase of cybercrime and cyber terrorism. Consequently, computer users have raisedthe level of anxietyin information security. The common question from any computer user would be 'How securely a message can be sent over the internet?' The level of security required by the user may vary from very low (such as forwarding a joke) to a very high (sending credit card information). To provide required security many attempts can be seen, and most widely used method is based on cryptography.

Cryptography is the study and practice of encoding datausing transformation techniques so that it can only be decoded by specific users. In simpler words, it is a theory of secret writing. Cryptography is accepted as the most secured method of sharing information by security experts. However it has its own inherited weaknesses. For an example, the eavesdropper can easily suspect that a secret message is been transmitted just by tapping the message. Hence the eavesdropper may use cryptanalysis techniques to reveal the original message. As a counter measure for this weakness, a technique is desirable which should not draw attention of eavesdroppers. When a secret message is been transmitted. This is the exact idea behind the concept of Steganography. The main advantage of steganography when compared to cryptography is that the eavesdropper would not suspect that there is a hidden secret message, and hence it may not draw their attention. While cryptography is about protecting the content of messages, steganography is about concealing their very existence[1].

The word steganography is of Greek origin and means "concealed writing" from the Greek words steganos (στεγανός) meaning "covered or protected", and graphein (γράφειν) meaning "to write". The first recorded use of the

* Corresponding author:
gvijayaratnam@bcs.org(VijayaratnamGaneshkumar)
Published online at http://journal.sapub.org/computer

term was in 1499 by *Johannes Trithemius* in his Steganographia, a treatise on cryptography and steganography disguised as a book on magic[2]. The first recorded uses of steganography can be traced back to 440 BC, when Herodotus mentions two examples of steganography in *The Histories of Herodotus.Demaratus* sent a warning about a forthcoming attack to Greece by writing it directly on the wooden backing of a wax tablet before applying its beeswax surface[1], and this believed to be where the steganography had started. From the day it started, steganography is performed on images, multimedia files, and text, word and PDF files. Time to time steganography technique has evolved, and modern techniques are performed on H.264 video sequence[3], power point files[4] and stegno-digital-signals[5]. As a practice, message is entered usually in only one language. Most of the steganography algorithms are language depended. Researcher *NatthawutSamphaiboon* have proposed a steganography method for Thai text[6], *Mohammad Shirali-Shahreza* proposed method for Persian/Arabic Unicode text[7] and *Changder* presents new technique for Hindi language[8]. However, in the proposed framework, it supports multi languages (mixture of languages that can be used in a single message).

2. Addressed Problem

In encryption, there are several algorithms such as RSA, Triple-DES, Blowfish, etc[19]. PGP and Microsoft are two main companies providing commercial cryptographic libraries which enable encryption. Therefore developers can use the API/SDK provided by them and develop their own security components. However, there is no such generic framework available for steganographic techniques.

Presently modern steganography targets the digital medium such as images, documents, media files (audio/video) to hide secret messages. For example in in the case of text based steganography. There are several algorithms like Line shifting and word shifting. However for an end-user, it is not that easy to use such an algorithms as there is no user friendly tools/software available in the market. This paper presentsa layered architectural framework which can be used by the developer as well as end-user. In the case of developer, Steganographic Framework supports component or module based development platform where they could develop framework modules. For the end-user, how stegno medium is generated is not that important but what they really need is an application that would generate stegno message with minimum effort. It should also facilitate such as flexibility, ease of use and high security.

According to the previous researches in steganography methods, each of them provides a language dependent algorithm, so that they cannot be used to send multi language messages. The proposed framework and the stegno RTF module is not language dependent and can be used in multi-language messaging and multiple language messages. In the case of multi-language message which consists of more than one language such as Sinhala, Tamil and Arabic. Hence it supports messages in different combinations of languages as follows:

- Combination of languages in one language.
- One message in one language, but collection of messages in different languages.
- Independent from the transmitted language.

3. Related Work

A. Steganography on Images

Image is the most popular stegno channel used in steganography. Image is a collection of numbers that constitute different light intensities in different areas of the image[9]. This numeric representation forms a grid and the individual points are referred to as pixels. Most images on the internet consists of a rectangular map of the image's pixels (represented as bits) where each pixel represents its colour[10]. The least significant bit (LSB) insertion is the most common way of embedding messages in the cover image. The least significant bit (8^{th} bit) of all the bytes in the images is encoded with the secret message.

B. Text Steganography

There are quite a number of researches had already explored in new textual steganographic techniques.

In *Line shifting* method, the lines in the text are vertically shifted to some degree (each line is shifted to 1/300 up or down) and the secret information is hidden by creating a unique shape of the text[11]. Also, if the text is re-typed or a character recognition program (OCR) used, the information will be destroyed.

Shifting the words horizontally and changing the distance between words, and then information is hidden in the text[11] are called *Word shifting*. Same as *Line shifting*, retyping or OCR will destroy the message.

C. Stegano on H.264 Video using Chaos-based Algorithm

Recently *BoWang*et. al. has done an exploration on how a message can be concealed into H.264 video stream. A steganographic algorithm for H.264 standard is proposed in their paper, in which chaos encryption is applied to the secret message before embedding[3].

D. Steganography on WLAN

Most of the network based steganography focus on Ethernet header modification and some focuses on timing channel based steganography[5].

The IEEE 802.11 MAC frame provided by the protocol has to be evaluated for possible points of transparent modification, where an embedding has no impact on the overall functionality of the underlying network traffic (the cover). This frame, which contains all IEEE 802.11 protocol data as well as the payload and consists of the nine fields[5].

E. Steganography in Thai Text

NatthawutSamphaiboon and *Matthew N. Dailey*[6] proposed a new blind steganographic scheme for Thai text that exploits redundancies in the way TIS-620 represents compound characters combining vowel and diacritical symbols.

They found that the modifications made, when information bits are embedded in the carrier text are unnoticeable to casual observers[6].

F. Persian/Arabic Unicode Text Steganography

Two Iranian researchers *Mohammad Shirali-Shahreza* and *SajadShirali-Shahreza* have proposed a steganographic method conceal Persian and Arabic text using the different shapes regarding to its position in the word[7].

G. Hindi Text Steganography

An Indian researcher*K.Prasad,* invented a steganographic algorithm to conceal *Hindi* letters and its diacritics and numerical code. He suggests his method of approach can be applied to Hindi like other Indian languages[12].

4. Part I – Steganography in English Language

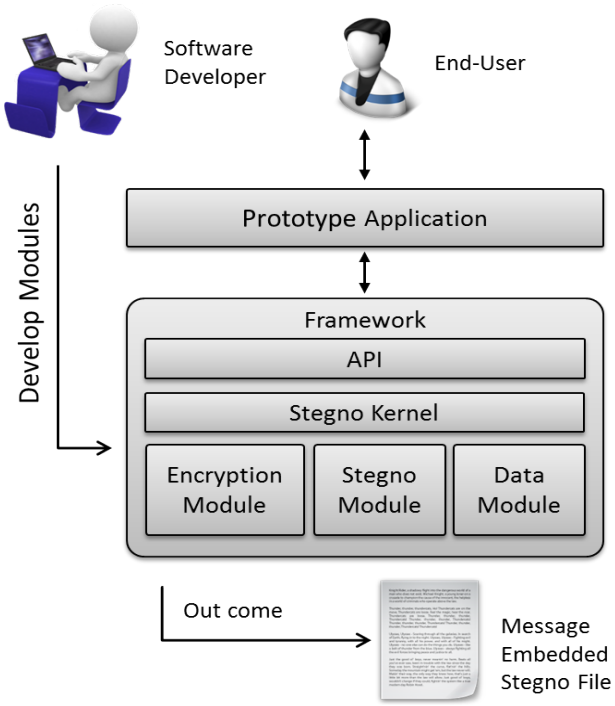


Figure 1. Steganography Framework

In this section primarily describes the overall usage of the framework from developers and end-user point of view. Pre-encoding technique which is an essential step before the stegno algorithm is applied, but after the encryption process. As the last section of this part, the stegno algorithm which uses the RTF document as the stegno medium and its embedding algorithm is desirable.

A. Introduction to Steganography Framework

Fig.1 describes the high level design view of the framework.

Framework has a user friendly API (Application Programming Interface) so that developers can easily develop different modules of the framework for end-users.

End user will execute the prototype application and its modules to generate steganographic messages. As the first step, user will enter secret message. Then to select the encryption module, if more security is required as it is optional. Next is to select the stegno module. However user can select which medium to be used to hide the message. Finally, the framework will generate the stegno file, where the message has been embedded. In the process of steganographic message generation using the framework end-user need not to know the technical implementation of the each stegno algorithm, but simply select it.

As for developers, they can develop framework modules by implementing the interfaces (which will be discussed later sections) according to defined standards. These modules could be *Stegno*, *Encryption* or *Data*, which can be distributed among end-users

B. Encoding using User Defined Codes (UDC)

There are three main aspects that need to be considered in the context of steganography: *capacity*, *security* and *robustness*[16].

Capacity refers to the amount of information that is able to be hidden in the medium, meaning Steganographic algorithm should be always capable of reducing the message size. This avoids number of alterations done on the stegno medium, in order fight against steganalysis techniques (method of detecting hidden messages in a given medium).

Security is not that important when a secret communication is kept to be a secret and undetectable by eavesdroppers. However, an additional security in steganography may be achieved by an encryption algorithm before embedding or some kind of custom coding mechanism.

Robustness can be explained as the amount of modification the stegno medium can withstand before an adversary can destroy hidden information[16].

User defined code technique is originally invented by *Potdar* using 5-bit encoding for each English characters[17]. His technique mainly focuses on *capacity* and *security* by pre-step before the embedding. This technique translates the message text data into a user defined 5-bit code, rather using 8-bit binary, before embedding algorithm is applied.

Table 1. UDC using 6-bit codes

Character	Code	Character	Code	Character	Code
a	000000	G	000110	m	001100
b	000001	H	000111	n	001101
c	000010	I	001000	o	001110
d	000011	J	001001	p	001111
e	000100	K	001010	q	010000
f	000101	L	001011	r	010001

Table 2. Special characters included in 6-bit UDC

!	@	#	$	%	?	&	{	[	*
(	)	]	}	-	+	=	/	“	‘
;	:	<	>	,	.	space		\r	\n

The proposed 6-bit encoding scheme extends his technique so that it can support alpha numeric characters and special characters.

Table 1 shows a few character mapping using 6-bit UDC. As *Potdar*[17] explained in his paper, numeric values are not represented in numeric format, but represented in textual format (i.e. 20 = twenty). As an extension to his method, the numeric values are directly mapped using 6-bit UDC, rather converting to textual format.

In 6-bit UDC encoding, it is possible to store up to maximum of 64 characters. Allocating 24 characters for the alphabet (a…z), and 10 characters for the numeric values (1…0), remaining 30 characters can be utilized for special characters. In general, a message consist not only alphabets, but also it can have numeric and few commonly used special characters like <Space> and <question mark>.

Table 2 shows some of the special characters included in the 6-bit UDC. UDC encoding directly supports this characters. However, *Potdar's*[17] method cannot support this as it has limited number of bits in 5-bit code.

C. Message encoding to UDC

Before the embedding algorithm is applied, the message will be converted to 6-bit UDC. For example if the word “*apple*” is encoded, then the result would be (Fig. 2):

000000	001111	001111	001011	000100
a	p	p	l	e

Figure 2. Encoded message

Once the message is successfully encoded, it is ready to be embedded into a stegno medium. It is notable that the proposed method, using 6-bit has advantages over *Potdas* technique.

At the first glance one would think, a 6 bit coding may consume more space compared to the 5 bit. It is true only for messages with only alphabets or English characters. However, for messages with numbers and special characters, it is other way round.

As an example, consider a message where value “20” is included. As per *Potdar's* method, value is first converted to English alphabet like “twenty”, and it has to be encoded into 5-bit code. This method requires 30-bits to store the value “20”, according to the following calculation.

$$6 \text{ characters x } 5\text{-bit} = 30 \text{ bits.} \quad (1)$$

However the proposed technique uses 6-bits to represent one character, and the numeric values are encoded as it is. Number of bits required can be calculated as depicts in Equation (1).

$$2 \text{ characters x } 6\text{-bit} = 12 \text{ bits.} \quad (2)$$

This is one drawback in *Potdar's* method when compared to the proposed technique. S, in the example, the proposed technique has used lesser number of bits compared to *Potdar's* method (E.Q 1 – 2), even though it had been encoded in 6-bits. In addition to that, the proposed method can support special characters, which *Potdar's* method could not. The detailed comparison of 5 and 6-bit is discussed in the experimental results section.

D. Steganographic Algorithm

For the proposed stegno algorithm RTF (Rich Text Format) file was used as the medium. RTF is a method of encoding formatted text and graphics, so that they can be easily transferred between applications. The RTF standard provides a format for text and graphics interchange, which can be used with different output devices, operating environments, and operating systems[20]. An RTF file consists of unformatted text, control words, control symbols, and groups. For easier transmission, a standard RTF file can consist of only 7-bit ASCII characters.

In order to embed the UDC encoded bit stream into the RTF file effectively, following command is used:

“\expnd” – Expansion or compression of the space between characters in quarter-points.

By using this section keyword, it is possible to increase the inter-character spacing of word. For an example, value “1” is embedded in the word “Steganography” would be **“{\expnd1 Steganography}”**.

In addition, the control word **“\qj”** is used to justify the paragraph, so that human eye cannot detect the inter-character spacing modification.

The Fig.3 shows the result after embedding the bit stream (Fig.2) in to a RTF file.

{\rtf1\ansi{\fonttbl\vertalj\f0\fswissHelvetica;}\f0\pard Geographical **{\expnd0 proximity}** and **{\expnd0 trade}** brought **{\expnd0 Ayurveda}** to Lanka **{\expnd0 some}** centuries before **{\expnd0 the}** birth of **{\expnd0 Christ}.** Yet, **{\expnd0 since}** time **{\expnd0 immemorial}**, and **{\expnd1 even}** before **{\expnd1 the}** advent of **{\expnd1 Ayurveda}**, indigenous **{\expnd1 deities}** have **{\expnd0 been}** invoked by **{\expnd0 indigenous}** medicine **{\expnd1 men}** to absolve **{\expnd1 man}** of physical **{\expnd1 and}** mental **{\expnd1 malady}**. **{\expnd0 Furthermore}**, geographical **{\expnd0 location}** has **{\expnd1 blessed}** the **{\expnd0 island}** with an **{\expnd1 abundance}** of flora **{\expnd1 containing}{\expnd0 valuable}** medicinal properties. **{\expnd0 In}** addition to these, **{\expnd0 Unani}** made its way to **{\expnd1 Lanka's}** shores with **{\expnd0 Muslim}** traders, as **{\expnd0 allopathy}** and homoeopathy came with western civilization. Immaterial the nature of disease, the right to be healed was the fundamental right of everyman.And the diseased was at liberty to use that system, or systems that guaranteed the greatest relief.\qj\par}

Figure 3. UDC Encoded RTF file

The bit stream is inserted using “**\expnd**” keyword in to randomly selected words in the RTF document. The raw view of the result is shown in Fig.3. Even though the proposed algorithm successfully inserted the secret message in to the RTF file using inter-character spacing, still it is undetectable to the human eyes. For the human eyes the result will look like in Fig.4. In the Fig. 4, the word ‘even’ (which

is underlined) has an inter-character spacing of one quarter-point between the characters. However for the naked eye it looks like the original and the change is not notable as human eye cannot differentiate such a small different.

"Geographical proximity and trade brought Ayurveda to Lanka some centuries before the birth of Christ. Yet, since time immemorial, and even before the advent of Ayurveda, indigenous deities have been invoked by indigenous medicine men to absolve man of physical and mental malady. Furthermore, geographical location has blessed the island with an abundance of flora containing valuable medicinal properties. In addition to these, Unani made its way to Lanka's shores with Muslim traders, as allopathy and homoeopathy came with western civilization. Immaterial the nature of disease, the right to be healed was the fundamental right of everyman.And the diseased was at liberty to use that system, or systems that guaranteed the greatest relief."

Figure 4. UDC Encoded RTF in word processor view

In addition, the proposed steganographic algorithm is not limited to English stegno RTF document. This means that the RTF embedding technique can be used with any language within RTF files. For example, an Arabic secret message can be embedded into a Sinhala RTF document. This increases the robustness, security and reduces the suspicions of the secret message.

E. UDC Decoding

This section will discuss the decoding process at the receiver's end which supposed to regenerate the original secret message. It's a straight forward method as follows. Once the bit stream is extracted from the RTF stegno medium, it is grouped in 6 digit blocks and mapped with Table 1 to retrieve the original characters.

F. Steganography Framework

This section elaborates via design of a generic steganographic framework that enables software developers to build their own steganographic modules. It also provides an interface so that the end user can easily generate stegno message with minimum effort.

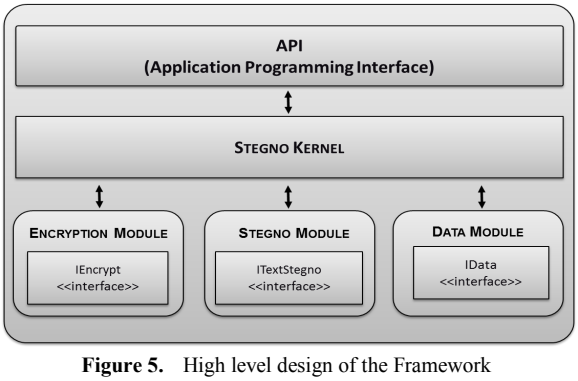


Figure 5. High level design of the Framework

As shown in the Fig.5, framework consists of four major modules; namely *Encryption, Stegno, Data modules and Stegno Kernel*. Each of these modules plays a vital role in generating stegno medium.

1) Encryption Module

As explained earlier, security is an essential part of steganography. Security of the secret message is achieved by the framework via well-defined interfaces. This interface is designed with synchronous and asynchronous encryption methods. An integrated PGP module preforms this task for this research.

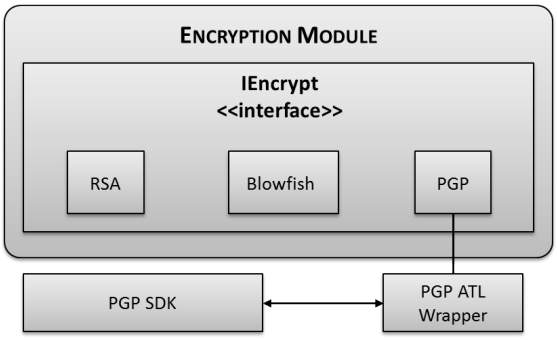


Figure 6. Encryption module

For the prototype purpose, PGP is used as the encryption module. This implements *IEncrypt* interface. Framework was written in .Net C# language, and a wrapper PGP ATL (Active Template Library) component was developed in C++ because, PGP SDK doesn't natively support .Net platform. Like PGP module, anyone could develop an encryption module, given that *IEncrypt* interface is implemented.

2) Stegno Module

The core module or the heart of the framework is the Stegno module which executes the information hiding algorithm. In section IV, it explains that the steganography can be used in many digital mediums, such as images, media files, documents and even in TCP/IP channels. As a steganographic framework design it should support all the stegno mediums.

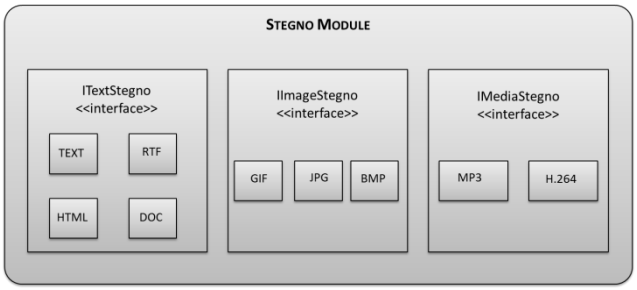


Figure 7. Stegno module interface

As shown in Fig.7, different types of stegno mediums are supported under different interfaces. For example if an image steganography module needs to be developed, *IImageStegno* interface must be implemented. Similarly each medium needs to use its corresponding interface. Proposed steganographic algorithm is based on RTF document, which is a text based file, and therefore this module should implement *ITextStegno* interface. Stegno algorithm in MP3 files and for RTF is not the same. The MP3 module needs an audio stream and the RTF module is with a RTF file. Because each medium is different from each other it is always preferable to have a separate interface for each category so that the development would be easier.

3) Data Module

Data module acts as a raw or stegno data provider to the

stegno modules. In order to successfully hide the secret message, each stegno module needs its own database. Data module will act as a database which means, the collection of text documents, images and all other digital medium. For example, a HTML stegno module hides the information inside a HTML file. For this, the stegno module needs dummy HTML document. This dummy HTML document will be provided by data module.

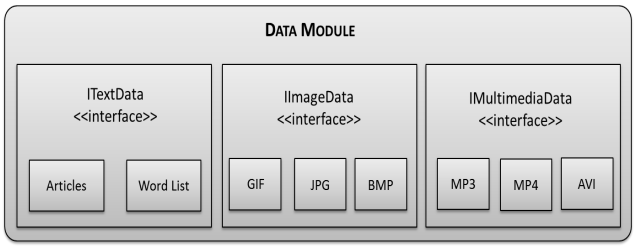


Figure 8. Data module interface

The Fig.8 shows the framework supported data module interfaces. As per this research, RTF file is used as the stegno medium, which is a text based file. For the prototype demonstration, *ITextData* interface is implemented in the data module which was developed to provide raw news articles. Similarly *IImageData*, *IMultimediaData* interfaces may be implemented by image or multimedia (audio/video) data module providers respectively.

4) Stegno Kernel

Stegno Kernel is the primary coordinator of all the interfaces and it is the most intelligence segment of the framework. In other words, it is the decision making component and it also handles the message routing. Depending on the users input, kernel will decide which plug-in modules are suitable to generate the stegno files.

Each module is identified with a unique id and by its content type supported. Table 3 shows few key properties of a module.

Table 3. Framework module properties

Property Name	Description
ModuleName	Name of the module. E.g: RTF Encoding or PGP
ModuleId	19 digit unique id E.g: 5407270172003320585
FileVersion	Version of the module.
ContentType	What kind of content it is supported, not applicable for encryption module. E.g: ContentType. TXT
Platform	Which platform it's supported. E.g: PlatformID.Win32 PlatformID.WinCF
Uli	Which languages are supported, English, Tamil or All Unicode languages.

Decision making of which module to be used, depends on the parameters as shown in Table 3.

Even though the proposed method (Part I), had advantages over the *Potdar's* technique. UDC encoding is tightly coupled with English language or this encoding is only supports English characters. Therefore, this research has been further extended to support on multi language encoding techniques using Unicode.

5. Part II – Multi Language Messages

According to the past researches, most of the steganographic algorithms are tightly coupled with particular language. These algorithms cannot hide languages. As an extension, this section proposes a language independent encoding method, which can encode messages in any language or combination of languages which can be represented in Unicode.

What is Unicode?

Computers deal with numbers in different bases (binary, octal, hex). They store all alpha numeric & special characters by assigning a unique number for each one. Before Unicode was invented, there were hundreds of different encoding systems for assigning these numbers. There was no single encoding system could contain enough characters: for example, the European Union alone requires several different encodings to cover all its languages. Even for a single language like English, no single encoding was adequate for all the letters, punctuation, and technical symbols in common use[13].

These encoding systems have conflict with one another. That is, two encodings can use the same number for two different characters, or use different numbers for the same character. Any given computer (especially servers) needs to support many different encodings; yet whenever data is passed between different encodings or platforms, that data always runs the risk of corruption[13].

Unicode provides a unique number for every character, which is independent from the platform, the program, and the language[13].

A. Enchanced Encoding using UDC

According to the Unicode standard, Unicode values range from 0000-FFFF, which contains 65535 characters. Thus each language has its own range of character mapping according to the Unicode standard.

Table 4. Few Unicode Character Range Mapping

Language	Range
Sinhala	0D80 – 0DFF
Tamil	0B80 – 0BFF
Basic Latin (English)	0000 – 007F
Arabic	0600 – 06FF
CJK – Extension A (Chinese, Japanese and Korean)	3400 – 4DBF

Table 4 shows few Unicode range mapping for chosen languages. The first two characters of the start and end range values are the same. E.g. for Sinhala, the first two characters are '**0D**'. Similarly,for most of the languages Unicode values are assigned. Neverthelessfor CJK (Chinese, Japanese and Korean) it is exceptional. The total number of Chinese characters from past to present remains unknown as new ones are being developed time to time. Chinese characters

are theoretically an open set[14], but Unicode supports over 35000 *Han*characters[16].

In the proposed method, first two characters of the range value are considered as the *Language ID* and the last two as *Character ID* as shown in Fig.9.

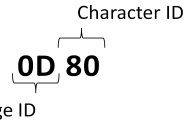


Figure 9. Custom language and character ID mapping for Sinhala language

As discussed in the previous section, special kind of encoding method is required to store not only Unicode values, but also any alpha numeric or special characters in a stegno medium. Consequently 3-bit UDC encoding is introduced for each Unicode characters. Unicode values are in hexadecimal numbers and hence it may vary from '0' to 'F'.

Table 5. Standard UDC using 3-bit codes

Hex Character	UDC
0	001
1	010
2	011
3	100
4	101
5	110
6	122
7	002
8	020
9	022
A	200
B	202
C	220
D	211
E	112
F	121

As shown in Table 5, it uses 3-bit code to represent one character which includes '0', '1' and '2'. Hence there are 27 combinations including '000'. In 27 combinations 17 were utilized for hexadecimal combinations ('0' to 'F').The other 10 are reserved for special character mapping, which are mostly used in any languages. This is discussed later sections.

B. Single Language Encoding

In general when a message or sentence is articulated, we use only one language. For example "Alice is waiting for Bob". This message is typed only using English language, like wise we may use Sinhala or Tamil. However, a mixture of languages can be found in certain situations. For example "Data" is translated "" in Sinhala. Most of the times, English term is used for technical explanation when a message is put into words in local languages. Local language can be more understandable if English term is used for better explanation. For example "(data)" or "தரவு(data)" the local language is more understandable since English is used in braces.

This section describes how to encode a message which was articulated using only one language using the proposed method. Let's assume a message in Sinhala ''is the secret message that needs to be encoded.

First step is to identify each character in the message and list down with the respective Unicode values and the corresponding *User Defined Codes* as in Table 5.

As described earlier (Fig.9), the first two characters of the Unicode values are used as *Language ID* and the last two as the *Character ID.*

Table 6 (column UDC) shows results of UDC encoding for each Unicode value.Since the entire message is in single (Sinhala) language, all the *Language IDs* are same for all the characters. Therefore, it is possible to assume the *Language ID* as the common value and excludes it from subsequent characters except for the first. The Fig.10 depicts the complete UDC encoded bit stream of the message.

Table 6. UDC using 3-bit codes

SinhalaCharacter	UnicodeValue	UDC
□	0DAF	001211 200121
□	0DAD	001211 200211
□	0DCA	001211 220200
□	0DAD	001211 200211

Please note that the *Language ID* (001222) is inserted only for the first character. At the receiving or decoding end, algorithm always assumes that the first six digits of the bit stream is the *Language ID*. Accordingly the decoding algorithm keeps on adding the *Language ID* with *Character ID* to get the correct UDC and map back with the corresponding Unicode character. As a result it can reproduce the same Unicode characters at the receiving end.

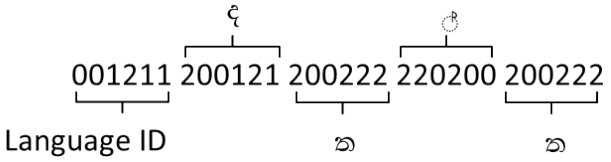


Figure 10. UDC encoded message

C. Single Language & Special Character Encoding

But it is not the same case if the message contains multiple languages in a single message. Some messages looks like single language; however, it may not.For an example note the following message written in Sinhala:

කෝ මගේ දත්ත?

Careful analysis would reveal that the message typed in combination of two languages; Sinhala and Basic Latin (English). The characters are in Sinhala but <SPACE> and

<QUESTION MARK> are in Basic Latin which belongs to a different Unicode range. These special characters are common for all the languages. It is interesting to note that there are few special set of characters which belongs to Basic Latin Unicode range but are widely used in many different languages. For example, the <SPACE> and the full stop (.) are the mostly used among them. To accommodate these commonly used special characters, an alternative way of encoding is being used.

As mentioned earlier in the previous section *(B)*, out of 27 combinations available, only 17 of them were utilized. Then the remaining 10 are allocated for those special characters directly. Table 7 shows the extended UDC to accommodate such frequently used special characters. UDC '111' is mapped to the <SPACE> character and '222' for (.) full stop.

Table 7. Extended UDC using 3-bit codes

Hex Character	UDC
<SPACE>	111
. (full stop)	222
, (comma)	212
?	221
<NULL>	000
(	012
)	021
' (single quote)	120
" (double quote)	210
- (hyphen)	102
!	201

In the above Sinhala message, the word '[illegible]' is encoded in the same way as discussed earlier. Next, the special character <SPACE> where the UDC value '111' is added to the bit stream. Below SimilarlyFig.5 shows the complete encoded message as explained.

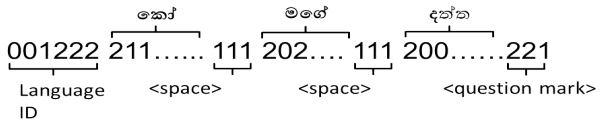


Figure 11. UDC encoded message with special character

D. Multiple Language Encoding

Now let's take a secret message with multiple languages (Sinhala, Tamil and English). As explained, the first six digits were used as the *Language ID* and then the rest of the *Character IDs* were added to the message as far as single language message concern. However, this would not work for messages with many languages. Consider the following message:

දන්න data தரவு

Figure 12. Multiple language messages

After the letter '[illegible]', the next character is in English, so then there must be new *Language ID* for the rest of the characters. At the receiving/decoding end, there must be a way to instruct the decoder of a language termination. Therefore, <NULL> character is used as the language terminator between two languages. Below Fig.13 shows the encoded message with language terminator.

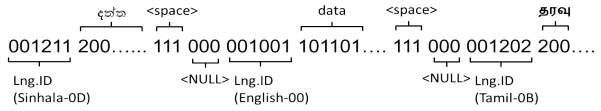


Figure 13. UDC encoded message with three languages

Whenever <NULL> character is found decoding bit stream, the decoder will consider the next subsequent six characters as a new *Language ID*, and then as usual it will go through the above mentions steps. Using this method any multiple language messages and any special characters can be encoded with any combinations.

E. User Defined Codes & Decoding

This section describes steps and the decoding sequence algorithm. Consider the following encoded message:

'001211200121200222220200200222111**000**001001122 101122010002101122010111**000**0012022001012020012021 10220101'

The below Fig.14 describes the decoding process in detail. As explained previous section, values are read in six digit blocks and the first six digits always represent the *Language ID*.

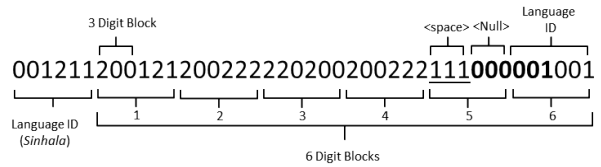


Figure 14. Decoding process

In the Fig.14, the first six digits are used to represent Language ID. Therefore the first block's (001 211) Unicode character is '0D' according to Table 5, the message starts with Sinhala (Table 4).

The block 2 (200121) is supposed to represent Character ID, but there might be special characters in the message. Thus the decoding algorithm read the next six digit block, and the, first three digits of it (200) are matched with the Table 7 to check if there are special characters. If there is no special characters all six digits are processed as one Character ID. But in this block there is no special character, so the 6 digit block represents the unicode value 'ODAF'.

In the block number 5 (111000), the first three characters are matched with Table 7. This time there is a special character, which is a <space>. Therefore to the decoded message a <space> character is added to the decoded message. With the remaining three digits plus the next three digits are grouped together to repeat the same logic (six block). Then again there is a <Null> which means a language terminator. A language terminator instructs the decoder that the subsequent message are in a new language. Therefore it should

consider the next six digit block as the new Language ID.So the next 6 digit block represents the English Language.

By applying these steps on the bit stream (Fing.14), will result to the correct decoding of the message successfully. Using this technique, it would be possible to encode and decode multi language and combination of languages in a single message.

6. ExperimentalResults

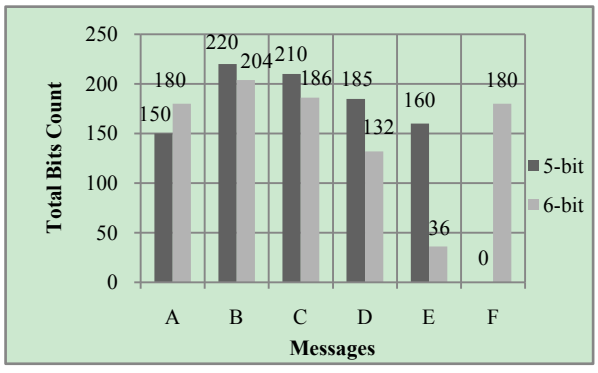
This section is separated into two parts. First part evaluates the 6-bit UDC encoding and framework, whereas Part II assesses the multi-language encoding using Unicode.

To evaluate the extended 6-bit UDC encoding, few sample messages were compared with *Potdar's* method. Table 8 shows the sample messages and the character count for each method (5-bit and 6-bit). At a glance one may think 5-bit is better as it gives higher compression; however there are advantages & disadvantages for each method.

According to the results (table 8), *Potdar's*method is better for the message 'A' as it takes only 150 bits compared to 6-bit method. This is because the message 'A' consists of only alphabetic characters. However, for message 'B', it is other way round, since numeric values are in the message. The *Potdar's* method requires more bits, because the numeric values are first converted to alphabetic characters (80000 → eighty thousand) and encoded. Same applies to messages 'C - E'.

Table 8. UDC encoding & comparison

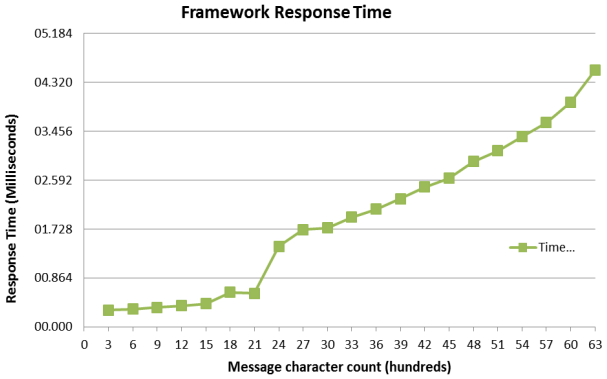
	Message	5-Bit(Character Count)	6-Bit(Character Count)
A	John Smith is a citizen of USA	150	180
B	I have bought a computer for 80000	220	204
C	I am getting 70000 as my salary	210	186
D	I have loan for 500000	185	132
E	350000	160	36
F	my email address is gk@msn.com	N/A	180



Graph 1. Visual representation of Table 8

Table 9. Number of characters vs. Framework process Time

Characters	Time(Seconds)
300	00.297
600	00.313
900	00.343
1200	00.375
1500	00.406
1800	00.610
2100	00.594
2400	01.422
2700	01.719
3000	01.750
3300	01.935
3600	02.078
3900	02.265
4200	02.469
4500	02.625
4800	02.922
5100	03.109
5400	03.359
5700	03.610
6000	03.968
6300	04.531



Graph 2. Visual representation of Number of characters vs. Framework process

The message 'F' cannot be encoded using *Potdar* 5-bit method, because the message has special characters, which his method does not support. But this can be encoded by the proposed 6-bit method, and it is a clear advantage compared to *Potdar's*method.

Potdar highlights that his aim is to keep the code as small as possible to increase the capacity of the transmission channel. His method is ideal for messages with only alphabets without numeric and special characters. However from the user point of view restricting them to input only messages without numeric and special characters is impractical. They will always want to send secret messages as they like, and need to use those. The proposed 6-bit encoding enables the user to enter messages as he/she types an email a document without limitation.

Framework response time evaluation graph is shows in Graph 2 and the respective values are in Table 9. Time is measured from the point where the framework started encryption, and then encoding to the final stage of creating the stegno RTF file. The test begins with 300 characters to process, which took 0.297 and for 6300 characters it took 4.531 seconds. The time taken to create a stegno message which requires 6300 characters to be hidden is a reasonable time. In addition to these response times, it is depended on

the individual modules, third party software like PGP which has been used.

For the next excrement, two types of sample messages were selected to evaluate the multi-language encoding technique.

Single Multi language message: In this case, the secret message is entered in using one language. Same message is interpreted into few other Unicode languages to show that proposed method can be applied to any Unicode based language. Consider the following message in English:

"multi language message"

Above message was translated into different languages such as Simplified Chinese (Table 11), Arabic, Hindi and Thai to evaluate the language independent encoding algorithm. In the encoding process, for the languages except CJK, first two characters of the Unicode values were used as the *Language ID*.

In Simplified Chinese, the *Language ID* is different for each character, and need special way of handling. Therefore, after each character, a combination of <NULL> and the new *Language ID* are used. Please note that, just for viewing purposes in the encoded message shown in Table.10, the NULL is marked in **BOLD** and the *Language ID* is underlined.

Multiple language messages: In this case, a secret message needed to be created using more than one language. In Fig.16, the shown secret message is entered in using more than one language, where each word of the message has been entered in a different language (Russian, Greek, and Urdu) to evaluate the multi-language capability of the method. Similar notation is used for clarity as in the above example:

Много ('multi' in Russian)
γλώσσα ('language' in Greek)
پیغام ('message' in Urdu)

Figure 15. Message in Russian, Greek and Urdu

Table 10. Message in Single Language (NULL is marked in BOLD and the Language ID is underlined)

	Simplified Chinese	**Arabic**	**Hindi**	**Thai**
Message	多语言消息	رسالة متعددة اللغات	**बहुभाषासंदेश**	**ข้อความภาษาหลาย**
Unicode Values	591A, 8BED, 8A00, 6D88, 606F	0631, 0633, 0627, 0644, 0629, 0020, 0645 062A, 0639, 062F, 062F, 0629, 0020, 0627 0644, 0644, 063A, 0627, 062A	092C, 0939, 0941, 0020, 092D, 093E 0937, 093E, 0020, 0938, 0902, 0926 0947, 0936	0E02, 0E49, 0E2d, 0E04, 0E27, 0E32 0E21, 0E20, 0E32, 0E29, 0E32, 0E2b 0E25, 0E32, 0E22
Encoded Message	110022 010200 **000**020202 112211 **000**020200 001001 **000**122211 020020 **000**122001 122121	001122 100010 100100 011002 101101 011022 111 101110 011200 100022 011121 011121 011022 111 011002 101101 101101 100200 011002 011200	001022 011220 100022 101010 111 011211 100112 100002 100112 111 100020 001011 011122 101002 100122	001112 001011 101022 011211 001101 011002 100011 011010 011001 100011 011022 100011 011202 011110 100011 011011
Transmitted Message in Raw View(Simplified Chinese)	{\rtf1\ansi{\fonttbl\vertalj\f0\fswissHelvetica;}\f0\pard Geographical {\expnd1 proximity} and {\expnd1 trade} brought {\expnd0 Ayurveda} to Lanka {\expnd0 some} centuries before {\expnd2 the} birth of {\expnd2 Christ}. Yet, {\expnd0 since} time {\expnd1 immemorial}, and {\expnd0 even} before {\expnd2 the} advent of {\expnd0 Ayurveda}, indigenous {\expnd0 deities} have {\expnd0 been} invoked by {\expnd0 indigenous} medicine {\expnd0 men} to absolve {\expnd0 man} of physical {\expnd2 and} mental {\expnd0 malady}. {\expnd2 Furthermore}, geographical {\expnd0 location} has {\expnd2 blessed} the {\expnd1 island} with an {\expnd1 abundance} of flora {\expnd2 containing} {\expnd2 valuable} medicinal properties. {\expnd1 In} addition to these, {\expnd1 Unani} made its way to {\expnd0 Lanka's} shores with {\expnd0 Muslim} traders, as {\expnd0 allopathy} and homoeopathy came with western civilization			
Transmitted Message in RTF View (Simplified Chinese)	Geographical proximity and trade brought Ayurveda to Lanka some centuries before the birth of Christ. Yet, since time immemorial, and even before the advent of Ayurveda, indigenous deities have been invoked by indigenous medicine men to absolve man of physical and mental malady. Furthermore, geographical location has blessed the island with an abundance of flora containing valuable medicinal properties. In addition to these, Unani made its way to Lanka's shores with Muslim traders, as allopathy and homoeopathy came with western civilization			
Decoded Message	110022 010200 **000**020202 112211 **000**020200 001001 **000**122211 020020 **000**122001 122121	001122 100010 100100 011002 101101 011022 111 101110 011200 100022 011121 011121 011022 111 011002 101101 101101 100200 011002 011200	001022 011220 100022 101010 111 011211 100112 100002 100112 111 100020 001011 011122 101002 100122	001112 001011 101022 011211 001101 011002 100011 011010 011001 100011 011022 100011 011202 011110 100011 011011

The result shown in Table.10 has proved that any Unicode language or combination of Unicode languages can be encoded and embedded successfully using the proposed method. Transmitted message is completely irrelevant to the secret message. Hence if unauthorized parties get access during the transmission, would notice only an unrelated message. However, the receiver can decode the received text and extract the original secret message by reversing the algorithm.

In Simplified Chinese, as the *Language ID* is different for each character, it is required to have a special way of handling. Therefore, after each character, a combination of <NULL> and the new *Language ID* are used.

As explained in previous section, there are three factors (Capacity, Security & Robustness) those are highly considered in the context of Steganography. In the proposed method capacity factor is achieved by pre-encoding the secret message, rather using 8-bit ASCII binary. By implementing the Encryption module, the framework achieves the Security aspect.

Table 11. Message containing three languages: Russian, Greek and Urdu

Message	Многоγλώσσα پیغام
Unicode values	043C, 043D, 043E, 0433, 043E 03B3, 03BB, 03CE, 03C3, 03C3, 03B1 067E, 064A, 063A, 0627, 0645
Encoded Message	001101 100220 100211 100112 100100 100112 *111* 000 001100 202100 202202 220112 220100 220100 202010 *111* 000 001122002112 101200 100200 011002 101110
Transmitted Message as RTF Raw View	{\rtf1\ansi{\fonttbl\vertalj\f0\fswissHelvetica;}\f0\pard Geographical {\expnd0 proximity} and {\expnd0 trade} brought {\expnd1 Ayurveda} to Lanka {\expnd1 some} centuries before {\expnd0 the} birth of {\expnd1 Christ}. Yet, {\expnd1 since} time {\expnd0 immemorial}, and {\expnd0 even} before {\expnd2 the} advent of {\expnd2 Ayurveda}, indigenous {\expnd0 deities}
Transmitted Message in RTFDocument View	Geographical proximity and trade brought Ayurveda to Lanka some centuries before the birth of Christ. Yet, since time immemorial and even before the advent of Ayurveda, indigenous
Decoded Message	001101 100220 100211 100112 100100 100112 111 000 001100 202100 202202 220112 220100 220100 202010 111 000 001122002112 101200 100200 011002 101110

7. Framework &Prototype Application Demo

This section describes the prototype application which was developed using Steganography Framework. This demo application was developed to prove the ease of use for the end-user. You may find an online video of the prototype application.

Demo: http://www.screencast.com/t/MAjp6dAcz

This prototype application uses RTF stegno module and PGP as the encryption module. Collection of news article database was used as the data module. The application development environment as follows:

.Net 2.0 and C#
Microsoft SQL server 2008
PGP desktop edition

8. Conclusions

A common steganographic framework is presented in this research paper is useful for both an end-user as well as the developer.The prototype application demonstrates how flexible it is for an end-user to generate a stegno medium. In addition, a third party security product integration support provides more secured stegno messaging. This was demonstrated by integrating PGP to the framework as an *Encryption module*. In this manner any security algorithm (RSA, Triple DES, and Blowfish) can be integrated as modules. As for developers, the framework architecture opens the creativity by providing *Stegno, Security and Data module* APIs, so that they can develop modules and offer to end-users.

UDC encoding is carried out as a pre-encoding step, to convert the secret message before applying the steganographic technique. This has further enhanced the strength of secrecy and compression of the information to be sent. In addition, the encoding supports multi language and multiple language stegno messaging. Therefore, an end-user can use this for messages containing any Unicode supported language and generate a stegno medium. This technique is further extends to encode multiple languages, so that a message can contain more than one language.

As inter character spacing of the RTF file is utilized as a steganographic technique, it is possible to hide a secret message in one language into a document in another language Consequently the embedding technique is not depended on the language of the RTF document (transmitting message) and hence even a third party revealed the secret message, it can be in a totally different language which they do not understand. Therefore the possibility of revealing the original message is very low. This even increases the robustness of the proposed method.

ACKNOWLEDGEMENTS

We would like to extend our very special heartfelt gratitude and appreciation to Mr. Marcelo Bossi, Senior Manager Business Development at PGP Corporation who has given extensive support by providing PGP SDK evaluation copy for our research, and also Mr. David Wiener, Associate General Counsel at PGP also remembered at this moment.

REFERENCES

[1] Stefan Katzenbeisse, Information Hiding Techniques for Steganography and Digital Watermarking, Fabien, Ed.: Artech, 99.

[2] Wikipedia.[Online]. http://en.wikipedia.org/wiki/Steganography.

[3] JiuchaoFeng, Guangzhou Bo Wang, "A chaos-based steganography algorithm for H.264 standard video sequences," Communications, Circuits and Systems, ICCCAS 08. International Conference.

[4] Ling-Hwei Chen Wen-Chao Yang, "A novel steganography method via various animation effects in PowerPoint files," IEEE, Machine Learning and Cybernetics, International Conference, vol. 6, pp. 3102-3107, July 08.

[5] Jana Dittmann, Andreas Lang, Tobias Kühne Christian Krätzer, "WLAN Steganography: A First Practical Review," ACM, International Multimedia Conference, pp. 17 - 22, 06.

[6] NatthawutSamphaiboon and Matthew N. Dailey, "Steganography in Thai Text", ECTI-CON 08.

[7] Mohammad Shirali-Shahreza, SajadShirali-Shahreza, "Persian/Arabic Unicode Text Steganography", The Fourth International Conference on Information Assurance and Security, 08.

[8] Changder, S. Debnath, N.C. Ghosh, D, "A New Approach to Hindi Text Steganography by Shifting Matra", International Conference on Advances in Recent Technologies in Communication and Computing, ARTCom '09, pp. 199-202.

[9] I.V. Karpinskyy, M.P. Sagan, A.M. Vasiltsov, "Development of VHDL-based core with embedded steganography function," CAD Systems in Microelectronics, CADSM 03. Proceedings of the 7th International Conference, pp. 260- 261.

[10] J.H.P. Eloff, M.S. Olivier T. Morkel, "An Overview Of Image Steganography," Proceedings of the Fifth Annual Information Security South Africa Conference.

[11] M., Tehran Shirali-Shahreza, "Text Steganography by Changing Words Spelling," Advanced Communication Technology, ICACT 08 10th International Conference, vol. 3, pp. 1912-1913.

[12] Alla, K. Prasad, R.S.R., "An Evolution of Hindi Text Steganography," Information Technology: New Generations, ITNG 09 Sixth International Conference, pp. 1577-1578.

[13] The Unicode Standard, http://www.unicode.org, last visited: 14 May 2010.

[14] Wikipedia,[Online]. http://en.wikipedia.org/wiki/Chinese_character#Number_of_Chinese_characters.

[15] Wikipedia,[Online]. http://en.wikipedia.org/wiki/Han_unification.

[16] Yong-Won Kim, Kyung-Ae Moon, and II-Seok Oh, "A Text Watermarking Algorithm based on Word Classification and Inter-word Space Statics," 03.

[17] V.M. Han, S. Chang, E, Potdar, "Dictionary Module and UDC: Two new approaches to Enhance Embedding Capacity of a Steganographic Channel," Industrial Informatics, INDIN '05. 3rd IEEE International Conference, pp. 697- 700.

[18] (2010, December) Tech-faq.com.[Online]. http://www.tech-faq.com/cryptography.html.

[19] (2010, Devember) Mycrypto.net.[Online]. http://www.mycrypto.net/encryption/crypto_algorithms.html.

[20] (2010, September) Tools.ietf.org.[Online].http://tools.ietf.org/html/rfc1896.

[21] PeterWayner, Disappearing Cryptography Information Hiding Steganography and Watermarking, 3rd ed.: Morgan Kaufmann, 2002.

[22] Ajith Abraham, Computational Social Network Analysis: Trends, Tools and Research Advances (Computer Communications and Networks), 1st ed., Aboul-Ella Hassanien, Ed.: Springer, 2009.

[23] Rainer Böhme, Advanced Statistical Steganalysis (Information Security and Cryptography), 1st ed.: Springer, 2010.

[24] Hiding in Plain Sight: Steganography and the Art of Covert Communication: Wiley, 2003.

[25] Stefan Katzenbeisser, Information Hiding Techniques for Steganography and Digital Watermarking: Artech, 1999.

[26] Jatinder N. D. Gupta, Handbook of Research on Information Security and Assurance, 1st ed.: Information Science Reference, 2008.

[27] Barry J. Blake, Secret Language: Codes, Tricks, Spies, Thieves, and Symbols: Oxford University Press, 2010.

[28] Dorothy Elizabeth Robling Denning, Cryptography and Data Security: Addison-Wesley, 1982.