# Text based steganography

Robert Lockwood and Kevin Curran*

School of Computing and Intelligent Systems,
Faculty of Computing and Engineering,
Ulster University,
Londonderry BT48 7JL, Northern Ireland
Email: lockwood-r@email.ulster.ac.uk
Email: kj.curran@ulster.ac.uk
*Corresponding author

**Abstract:** Steganography is the art of hiding information within other less conspicuous information to prevent eavesdropping by way of hiding its existence in the first place. Image based steganography is the most common form but text based steganography can also be used. Text based steganography can be generally classified as format based, linguistic and random/statistical generation. In general, random and statistical generated methods create a cover text but do not necessarily make semantic sense; that is, the subject matter of each sentence has little or no relation to the next sentence. Linguistic steganography can use natural language processing to hide information but again is still subject to analysis particularly if the basis for the cover text is an existing document. Here, we examine the leading methods of text based steganography. We evaluate a variety of steganographic techniques including open space encoding, synonym replacement, UK/US English translation algorithm and Wayner's Mimic Functions using five benchmarks which compare speed, capacity, complexity, compromisability and size. We find that the best methods to hide information should not use a single scheme, but a hybrid of many schemes. In order to further hide information, text should be compressed, encrypted and then hidden in a cover document.

**Keywords:** steganography; text based steganography; cryptography; security.

**Reference** to this paper should be made as follows: Lockwood, R. and Curran, K. (2017) 'Text based steganography', *Int. J. Information Privacy, Security and Integrity*, Vol. 3, No. 2, pp.134–153.

**Biographical notes:** Robert Lockwood is a graduate of Computer Science from the Ulster University. His research interests include text based steganography systems.

Kevin Curran is a Professor of Cyber Security and Group Leader for the Ambient Intelligence and Virtual Worlds Research Group at the Ulster University. He is also a senior member of the IEEE. He is most well-known for his work on location positioning within indoor environments and internet security. His expertise has been acknowledged by invitations to present his work at international conferences, overseas universities and research laboratories. He is a regular contributor on TV and radio and in trade and consumer IT magazines.

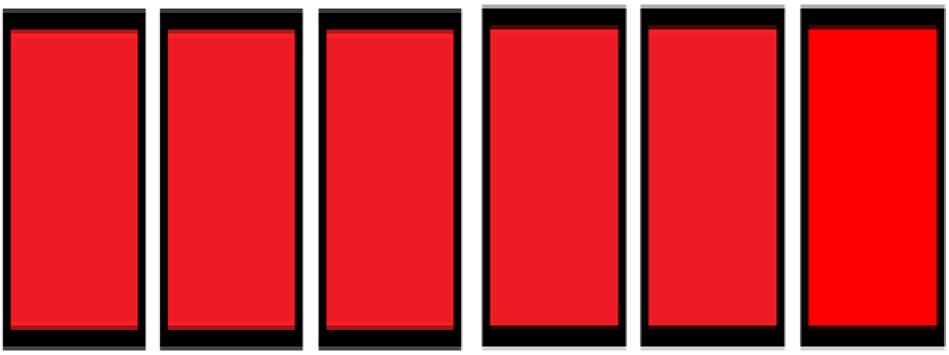
## 1 Introduction

Encryption of messages is now a common occurrence (Gupta et al., 2016). Popular applications include messaging, email and website queries. Whilst we feel fairly secure in the knowledge that encryption takes place, the very existence of the encryption can alert network peers, rogue routers and so forth to the presence of hidden information. Steganography is the art of hiding information inside a carrier such as an image, a sound file or network packets. The field of steganography has had much research especially with image based steganography but lesser research has taken place with text based steganography. Beyond email and watermarking, steganography has not become mainstream, yet the purpose of steganography is not to secure information as encryption but to hide its very existence in the first place. The origins of steganography was first coined by Trithemus who coined 'steganographia' which means 'concealed writing' (Bennett, 2004). Today steganography has been extended to not only include text but also images and any other object. For example, text can be embedded in images, video or other objects and vice versa with enough data to hide information in steganography can fall into five categories: images, video, audio, text (Bhattacharyya et al., 2010) and other objects such as executables which does not fit into the four original categories that Bhattacharyya described.

In general no matter the cover medium, steganography can be classified into two areas; key based systems and keyless based systems. A key based system hides information in a cover medium and generates a key for transmission on a separate channel. Only the sender and target receiver are aware of this key which would be used to expose the hidden information in a cover material. Keyless systems employ only the insecure channel to transmit and receive information but the sender and the receiver must be aware of the encoding algorithm in order to decipher the original information (Atawneh et al., 2016).

Image based steganography is *usually* the process of hiding text in an image by various means without distorting the picture noticeably to the user (Li et al., 2017). Other information can also be inserted such as other images. Significant research has taken place in this area (Bennett, 2004) and as such a brief overview of the most common methods will be explained. Some Image based methods do not employ modification of the image itself but can the file container in which the image is stored. One such scheme shown by Cheddad et al. (2010) explains that files can be appended to the EOF marker to hide data. Whilst this is ultimately very simple to implement for a small amount of information an image file significantly larger than the expected file size for the resolution may raise eyebrows and in itself cause further investigation. Certain Image formats also have areas within the format to hide small amount of data such as the EXIF field in images. Various research papers have used the encoding of data within the least significant bits (LSBs herein) within the pixels of the cover image. For example (Rig and Tuithung, 2012) also shown that the letter A can be coded into 3 pixels using the 3 LSBs of each pixel (3 BPP × 3 pixels = 9 bits which is enough to cover the 8 bits of the letter A). Figure 1 shows 3 pixels one without encoding and one with the letter 'A' encoded (zoomed).

**Figure 1** Original and difference encoding 'A' (see online version for colours)



As you can see this method cannot easily be identified by a person simply examining the image with their eyes. A steganalyst could detect the hidden data however if the image was significantly malformed which could arise when one attempts to insert too much information. Detection of hidden information is easier if one has the original image and is able to directly compare to the cover image. Given (in this case) a 1,024 × 768 image, using 3 pixels per character, 262,144 characters can be encoded or squashed together to form 294,912. Given that much of the ASCII character set is unused, a way to convert more information into fewer pixels would be to use of a custom character set that omits unused characters. Rig and Tuithung (2012) does this by way of Huffman Encoding. In the case of (Rig and Tuithung, 2012), they modify the DCT blocks of pixels in JPEGs but in essence any format can be used to encode information such as within bitmaps. The frequency of the characters being used form shorter bit lengths (such as 'A'). The letter 'Z' would less often be used so is located at the bottom of the binary tree and thus has a longer bit length.

Other methods of encoding information into images can be by manipulating the way the file is formatted by itself (Yu et al., 2017). Rig and Tuithung (2012) notes JPEG uses DCT blocks of 8 × 8 pixels as a form of compressing pixels and near pixels. Beyond JPEGs, different solutions can be applied to PNGs and other types. Videos on their very size make an attractive alternative to extremely large amounts of information in. For small amounts of data video based steganography would take a considerable amount of computational time (Balaji and Naveen, 2011) and network bandwidth, however, it can be suitable for large amounts of information. Depending on the format data can be held in frame by frame (within the pixels of the frame). Videos have another dimension in which information can be held, time. As with image based steganography, individual frames (which are images in their own right) can also be modified by changing the LSB pixels of the frame. As this has already been covered in image based steganography, it will not be repeated here as the concept is the same. Videos are divided into set of frames. Video formats can fall into one of two categories and some video formats support both: CBR and VBR. Beneath that frame rates can also vary. On high frame rate video formats, a single frame can contain a hidden frame. Due to the way our eyes work if the colour is nearly matching the rest of the frames the watcher would not notice. Whilst it can be used in steganography, it has also been used in subliminal messaging.

Steganography can take place in other objects and in theory any object. Executable files for the most part can also hide data and often do. Executable files do not necessary harbour the main application program itself but in some cases viruses, spyware and adware also. The Microsoft portable executable not only has sections for code (.text/.code segment) but data also; such as strings. Images are often included to form icons or *embedded resources* that are embedded into the application without having the resources externally stored. To the user the embedded content is hidden but exposable by using a resource extraction tool. Al-Nabhani et al. (2010) propose the use of header field of the portable executable. Immediately after the header, the hidden information would be stored. By updating the offsets of the starting program code, data and text segments, the capacity is high. Al-Nabhani et al. (2010) do note, however, in order for the application to execute it must first be downloaded and run (or installed and run). Like images and video, the least significant bits of audio data can also be modified. Because of the minimal modification to the generated sound, to human ears no distortion is identified. As with all plain LSB methods, steganalysis can potentially uncover hidden information. To overcome this (Asad et al., 2001) proposes selective modification of lower bits depending on the value of the most significant bits. This would make steganalysis more difficult but not immune particularly of the steganalyst is aware of the algorithm.

The audible range of human hearing is 20Hz (cycles per second) to 20KHz (Cuttnel and Johnson, 1998). Outside if this range humans cannot hear. If a significant safety margin is applied, we can encode audio below and above this range. In fact Gopalan and Wenndt (1998) did just this, while the selected frequencies in use are not outside of the audible range, low frequencies were used with the cover audio on top. Gopalan and Wenndt (1998) noted that this method is susceptible to noise and can cause the method to fail, any lossy medium could also cause data loss. Hiding videos in audio, images and text is possible but impractical because of the large size requirements. Images and audio can be encoded in such a way that they can be embedded into text with enough cover text, but most of all, all information can be interchangeable (McBrearty et al., 2017).

This paper provides an overview of text steganography and presents the stegaid text steganography system in addition to evaluation common text steganographic methods. We evaluate a variety of steganographic techniques including open space encoding, synonym replacement, UK/US English translation algorithm and Wayner’s Mimic Functions using five benchmarks which compare speed, capacity, complexity, compromisability and size.

## 2 Text based steganography

This form of steganography has had lesser research with comparison of other methods such as images, therefore is the focus of this research. *Steganographia*, literally means ‘covered writing’ (Bennett, 2004), and although this has now been extended to include images and other formats, the origins of steganography involve text. Text Based Steganography can fall into one of three categories: format based, linguistic and random/statistical generation (Bhattacharyya et al., 2010). In this chapter, we uncover the basic methods that can be applied to any language and not a specific language such as Chinese or Indian.

### 2.1 Format based systems

Format based steganography relies on a selected cover text and changing properties within the cover text such as punctuation, or spelling to hide information. More commonly information can be held in white space and non-printing characters.

The use of punctuation has been suggested as a way of hiding bits. Commas and full-stops are explored by Agarwal (2013). By selecting appropriate points of insertion of punctuation bits can be represented. For example, a full stop might represent 00, comma 01, exclamation 10 and question mark 11. Whilst, if the punctuation is logically correct, there is no reason to believe such an algorithm would ever be discovered.

Shirali-Shahreza (2008) propose the way in which words are spelled is a method in which to hide information. For example, in UK English the word 'favourite' and US 'favourite' have the same meaning but is inconspicuous in that the difference could represent a zero or one. Whilst this method is very simple, a selective approach to chosen words would have to take place as there are more English word spellings in common than different. In order to encode a large message, it could be estimated you get one bit per word, eight words to make a single character. On its own, the method would not be feasible for a significant amount of text due to the low encode result. If the method was combined with other methods to form a hybrid more bits can be encoded.

In some documents (namely HTML), spaces are ignored as are carriage returns. For example in order to structure the document <p> and <br> tags are used. The first space is accepted, but any additional spaces are explicitly ignored by the user's browser. In order to overcome this, website developers have to encode the &nbsp; code. Indeed (Barilnik et al., 2007) has explored this and whilst not suitable for normal documents can be used to hide information in source code, html documents or anywhere formatting is ignored and justified text. This can be useful for hiding a signatures (a form of watermarking) for copyright or hidden data. To a trained eye, this can be easily be subjected to identification through steganalysis. Barilnik et al. (2007) also noted that opening a HTML document in Microsoft Word and enabling the formatting marks, spaces are easily identified.

Other ideas include the colouring of white spaces (a hybrid method of the open space method above and applying colour). Naturally a white space coloured red on a white background is still white. Consequently for each space 3 bytes of information can be encoded (R, G and B, 1 byte each not including a possible Alpha channel).

Documents such as ODF and DOCX store document formatting by modulating the line spacing (by tiny amounts) to encode bits. Jalil and Mirza (2009) explains line shifting by a small amount of pixels can be used in watermarking as a form of document protection. Such a method can also be used in steganography to hide small amount of information or as part of a larger strategy in combination.

For printed material and direct to screen there are a number of using fonts to hide information. For some documents such as Microsoft Word, the user can easily see where the font changes by looking at the font field in the top icon bar. In HTML this could be used as viewers are not immediately aware of the font change. The following example contains two x characters, one bit 0 encoded and the other bit 1.

| | |
|---|---|
| hh | Hello World! How are you today? |

We can easily see the difference, but as part of a larger sentence it is difficult to pick up the differences. A computer application that can detect pixel level differences would easily pick up on these differences. Whilst these images are enlarged for visibility, what you may not notice is the additional character encoded in the 'Hello Wor'. In this case the letter is 'H'. Whilst un-zoomed and appropriately encoded, it is not immediately obvious. On paper, a pixel can be very small, for example a 600dpi Printer, 1dot equal 1/600th of an inch. The above methods are all extremely simple and whilst can either on their own or by using hybridisation are all candidates for steganalysis. The more encoding schemes employed, the more difficult it is to decipher the hidden information. In these cases the cover text can already exist but must be modified to hide the information in.

## *2.2 Random and statistical generation*

The second category of text based steganography involves generation of a cover text based on either randomisation or likelihood of correctness. This differs from linguistic based steganography which attempts to create a valid natural language text using a range of algorithms. This area borders the realms of computer science and language by way of natural language processing and computer generated texts.

Hernon Moraldo (2012) suggests the use of generation of a cover text hiding the information by way of Markov chains. Markov chains are often used to generate language based on words, bi-grams and so forth). Hernon Moraldo (2012) uses such a method to create a Markov chain based on a cover text. In the example provided, the book 'War and Peace' is used as the Markov generation source. It is noted, that whilst unigram based steganography is low quality, bigram generated texts are better. Despite this, it can still be identified there are issues as in the following example:

> "Be a square for fuel and kindled fires there. Secondly it was hard to hide behind the cart and remained silent. He feels a pain in the now cold face appeared that the man continually glanced at her as though they stumbled and panted with fatigue. With a deep."

Certain words are out of context in that "it was hard to hide behind the cart and remain silent". In this case, this approach can fall under linguistics and statistical generation.

Wayner (1991) created the well-known mimic functions and has been cited by a large proportion of text steganography research papers. He describes the process of the generating context free grammar using his mimic functions which are based on probabilities. Whilst the generated text is legible, the result does not make sense as in the following example.

> "Paul is dead! I am the walrus! Buy something right now. Do not shoplift. Buy! Buy!"
>
> "Here are the plans to the Overthruster, Sergei. Yoyodyne forever."

## *2.3 Linguistic steganography*

Linguistic steganography is the third major category, this involves the creation of a natural language text (or modification of an existing text) in order to hide information.

Coupled with a thesaurus, where the sender and receiver have the same thesaurus. Words based on existing text can be modified to represent values. This was proposed by Topkara et al. (2006) and others to act as a watermarking technique to protect documents. It has come to the attention of this researcher that you can have multiple related words. A proposed method could be to use a thesaurus and based on the value of the word an alternative is used. For example 'cat' == 'feline'. This could be extended further, there are 16 words or more (synonyms) for cat:

> bobcat cheetah cougar jaguar kitten leopard lion lynx panther puma puss pussy
> tabby tiger tom tomcat

In order to put things in context the thesaurus would have to have 'context'. Topkara et al. (2006) notes that a generated sentence may lose its context and may not make semantic sense. You'll note the following sentence contains a syntactically correct original sentence, a possible flawed sentence and a steganographic sentence:

> "My pet cat has been to the vet today."

> "My pet tiger has been to the vet today."

Tomcat is tagged with 'cat', 'pet' and 'tiger', whilst tiger is still valid, it may raise some eyebrows. Tomcat is the 16th word (0b10000). Reverse lookup suggests (with some computation) tomcat is a synonym for cat.

The above method would work if the thesaurus on the sender and receiver are identical, and assuming we have a thesaurus capable of identifying 'is a' relationships reverse lookup is not a problem. Thinkmap Inc. (2013) has such an application that shows relationships between words such as 'is part of', 'pertains to' and more specifically 'is a type of'. Other methods can also be used such as negativity (antonyms) of positivity to represent bits also.

A variation on synonym replacement is to replace the whole structure of the sentence. The previous method focuses on the modification of words (either synonyms or antonyms), this method requires use of natural language processing tools. Chang and Clark (2010) proposes the use of Google n-gram data to verify the correctness of the sentence although they have not proposed any type of medium in which the cover text be a basis for steganography. They have suggested news articles, but comparison with the original article and the modified text would yield suspicion. If we examine the following sentence:

> "The beginning of this month."

The sentence can be modified whilst still meaning the same thing:

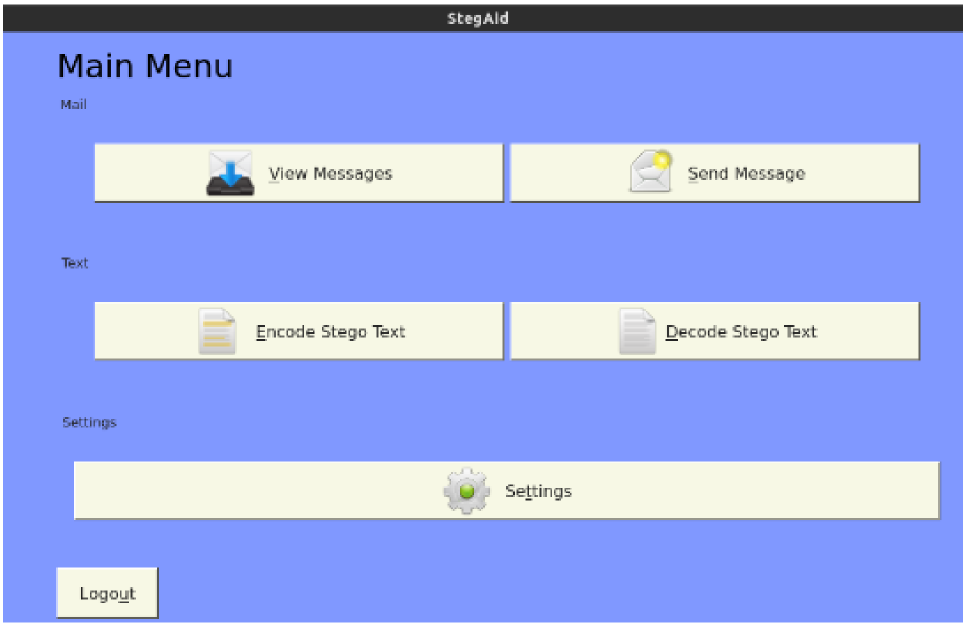
> "This month in the beginning ..."

The sentence is correct but has bits (zeroes and ones) encoded depending on its structure. This method is the simplest whilst remaining understandable to us. The issue arises when, if based on an existing article such as a news story, differences would be noticeable.

## 3 System implementation

We developed a tool for examining many popular text based steganography algorithms. After the splash screen has loaded (see Figure 2), the system prompts for user credentials. After the failure of input of credentials three times, the application will terminate. If authentication is successful, the user is passed through to the main menu interface. Specifically the password has hidden characters for security. A guest mode button appears if guest mode is checked in the system settings. If successful authentication occurs within three attempts, the main menu displays. The main menu allows for the sending and receipt of mail, the encoding and decoding of files and access to settings. If you belong to a group elevated permissions you will see the settings area, likewise, if you do not have access to mail, these options will also not be visible. In the example provided in Figure 2 the user has full administrative privileges. The buttons stand out significantly as users felt they could not identify these controls. Icons were also added as appropriate (along with the original planned shortcuts).

**Figure 2** Implementation – main menu (see online version for colours)



## *3.1 Encode text*

The primary capability of Stegaid is to encode and decode data using a variety of formats by way of text based steganography. Four selected methods currently exist using the plugin architecture. To hide information, the encode button can be selected (see Figure 3). At first all controls are hidden and activated only as appropriate to prevent confusion to the user, although a HTML Help File is also provided. Firstly the user should select a file to be hidden, be it some text, an image or other document. The only way to enter file is to click 'select file' which will provide the user with an open file dialogue.

Upon selection of a file, more controls become active. All registered libraries (plugins) are shown using a radio option select. If an option has been selected, an 'encode' button appears. Once encode has been selected, the core application, (Controller) loads the selected plugin, encodes the information before returning with the result and status messages. Encoding provides additional information such as the time taken to encode a file which can be useful for evaluation algorithms. In order to validate the encoding was successful, the performance timer is stopped and the cover text decoded and validated against the original file. There is currently no way to disable this feature and is considered essential to inform the user when things fail. Also some algorithms have capacity limits, these checks also occur.

**Figure 3** Implementation – encode text file selection (see online version for colours)

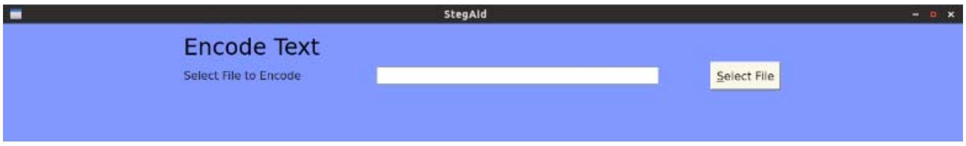
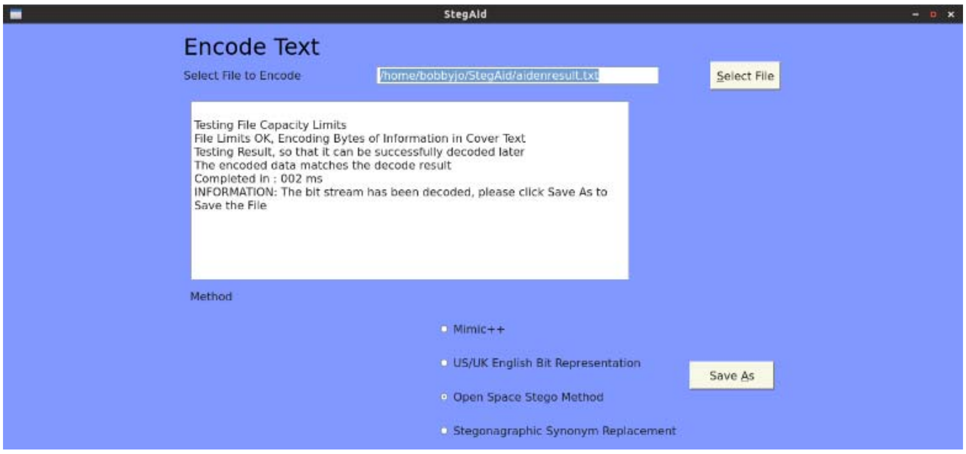


**Figure 4** Implementation – encode text status (see online version for colours)


StegAId
Encode Text
Select File to Encode
Select File
Method
○ Mimic++
○ US/UK English Bit Representation
○ Open Space Stego Method
○ Stegonagraphic Synonym Replacement
Save As

/home/bobbyjo/StegAid/aidenresult.txt

```
Testing File Capacity Limits
File Limits OK, Encoding Bytes of Information in Cover Text
Testing Result, so that it can be successfully decoded later
The encoded data matches the decode result
Completed in : 002 ms
INFORMATION: The bit stream has been decoded, please click Save As to
Save the File
```

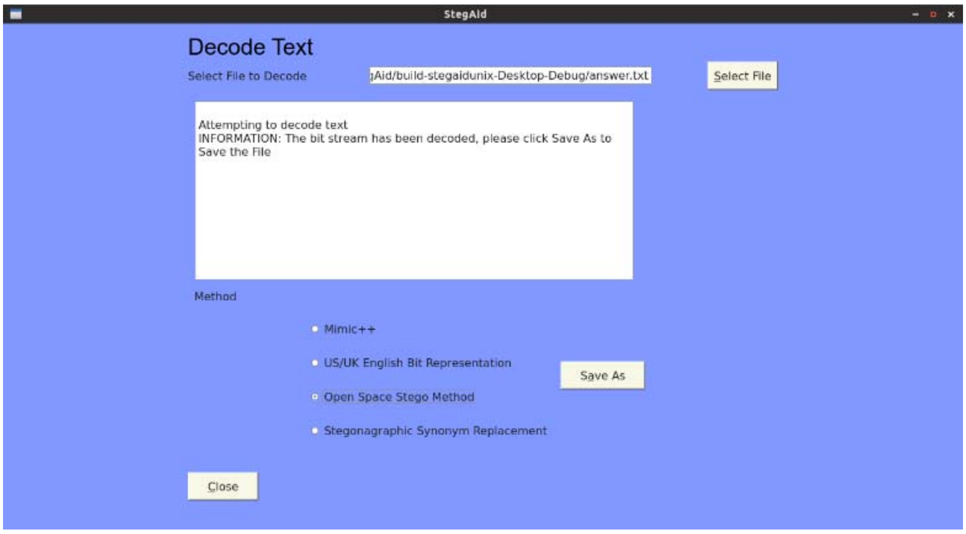
After encoding has taken place (see Figure 4), all invalid controls become disabled, and a ‘save as’ button becomes activated. The user can then activate this button to show a ‘save as’ dialogue and save the resulting cover text. Upon success confirmation is provided as to the status of the save process.

## *3.2 Decode text*

The decode text, opens a cover text and attempts to decode as appropriate. Like the encode text function, a stepwise approach to information elicitation from the user takes place to help avoid confusion to new users. After selecting the appropriate option from the main menu a decode text display is provided. After a file has been selected the additional options become active. The user must know how the file was encoded. Most steganographic systems have a single method, Stegaid has many. Consideration was made to placing a signature or embedded header into the cover text, however, this approach was not adopted in this release as it was felt a security risk to put implementation details of how a file is encoded into the cover text. If an incorrect method is selected, the decoding process will still occur, except the returned bytes of information will differ from the originally encoded data.

After the file selection and appropriate method selected the user can save the result file be it an image, or document. If the result was successful confirmation is provided as appropriate (see Figure 5).

**Figure 5** Implementation – decode text status (see online version for colours)



gAid/build-stegaidunix-Desktop-Debug/answer.txt

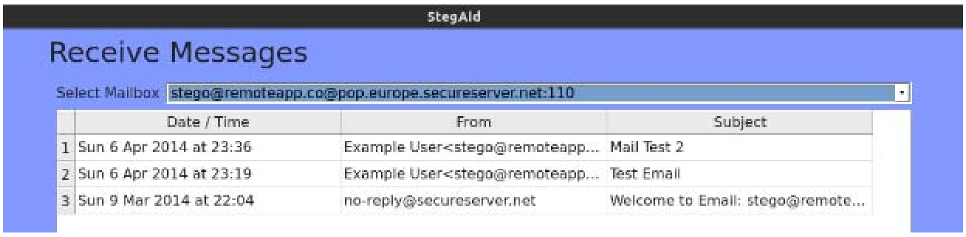
Attempting to decode text
INFORMATION: The bit stream has been decoded, please click Save As to Save the File

○ Mimic++
○ US/UK English Bit Representation
● Open Space Stego Method
○ Stegonagraphic Synonym Replacement

## 3.3 Receiving encoded messages

The Stegaid application toolkit provides functionality for encoded communications. Currently through the use of support libraries, SMTP, SMTP/TLS, POP3 and POP3/TLS email protocols are supported.

**Figure 6** Implementation – receive messages (see online version for colours)



Select Mailbox stego@remoteapp.co@pop.europe.secureserver.net:110

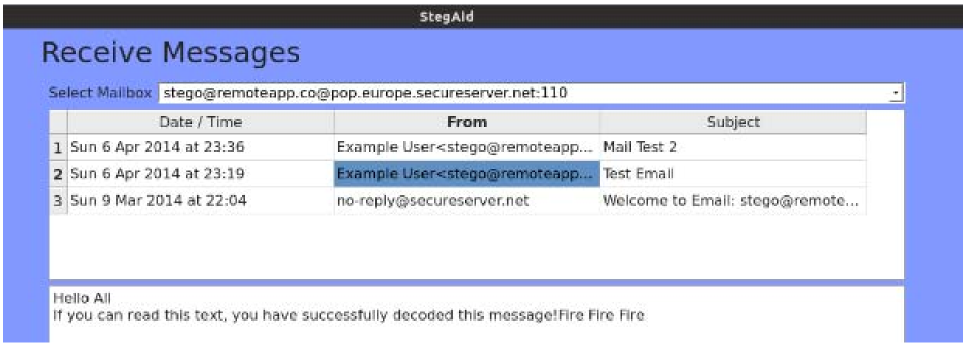
| | Date / Time | From | Subject |
|---|---|---|---|
| 1 | Sun 6 Apr 2014 at 23:36 | Example User<stego@remoteapp... | Mail Test 2 |
| 2 | Sun 6 Apr 2014 at 23:19 | Example User<stego@remoteapp... | Test Email |
| 3 | Sun 9 Mar 2014 at 22:04 | no-reply@secureserver.net | Welcome to Email: stego@remote... |

In Figure 6, we can see that currently no message has been selected, so no operations can be performed, hence no options have been enabled. Once a message is clicked relevant options become available.

The transformed display next shows once such decoded message. As a rule messages are downloaded but not deleted, this is intentional as Stegaid only supports encoded messages. Multipart MIME type messages are listed as un-encoded messages as they were not encoded by Stegaid.

If the user does not have any mail accounts set up but does have mail permissions, then a message box is displayed as soon as the display is loaded. As well as receiving encoded messages, mail messages can be encoded and sent using a compatible mail server such as Gmail and Go Daddy. As with the receiving of messages, the system checks for presence of mail accounts for the current user (along with other tests). This is to prevent the user typing a mail message only to find out that the message cannot be sent. Upon the successful sending of the message a confirmation message is displayed as appropriate. To manage the system and enable some customisability a 'settings' feature was implemented. This section has been locked down to any Group that has the appropriate permissions. Using a tabbed interface, items are in related groups, such as 'general settings', 'groups', 'users', 'libraries' and 'mail'. The general settings tab enable the ability to disable the console, allow for external authentication using a data plugin, enable mail and guest mode. Significantly there is no 'save' button, save occur internally either when the display is closed or a tab switch takes place.

**Figure 7** Implementation – received message view (see online version for colours)



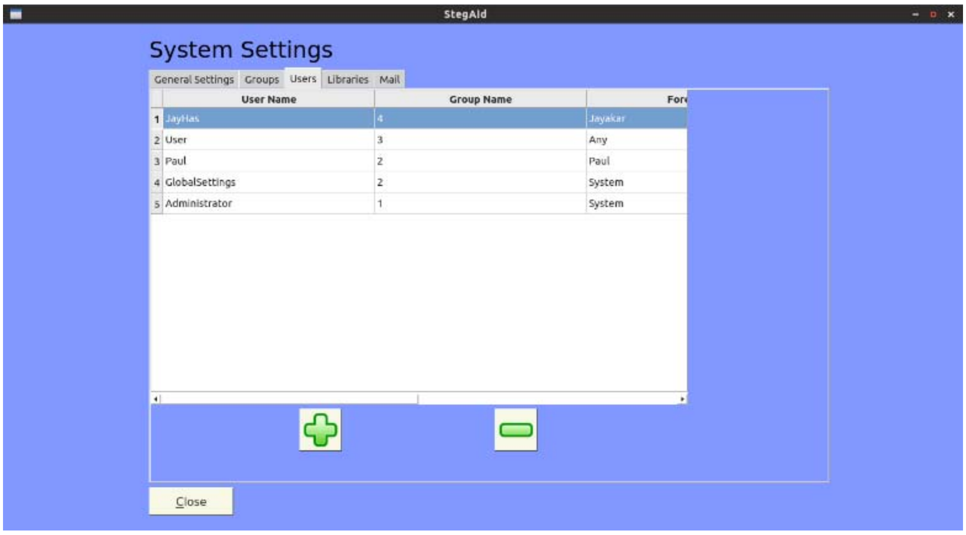
Receive Messages

Select Mailbox: stego@remoteapp.co@pop.europe.secureserver.net:110

| | Date / Time | From | Subject |
|---|---|---|---|
| 1 | Sun 6 Apr 2014 at 23:36 | Example User<stego@remoteapp... | Mail Test 2 |
| 2 | Sun 6 Apr 2014 at 23:19 | Example User<stego@remoteapp... | Test Email |
| 3 | Sun 9 Mar 2014 at 22:04 | no-reply@secureserver.net | Welcome to Email: stego@remote... |

Hello All
If you can read this text, you have successfully decoded this message!Fire Fire Fire

**Figure 8** Implementation – users (see online version for colours)

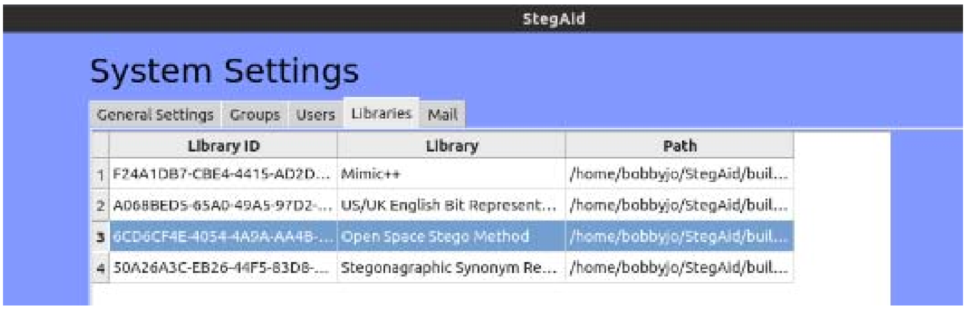


| | User Name | Group Name | For |
|---|---|---|---|
| 1 | JayHas | 4 | Jayakar |
| 2 | User | 3 | Any |
| 3 | Paul | 2 | Paul |
| 4 | GlobalSettings | 2 | System |
| 5 | Administrator | 1 | System |

Groups are logical grouping of users that have shared permissions. A certain group may have access to mail but not system settings, likewise, another Group may have access to no mail or system settings. Group names must be unique and are case insensitive along with other validation requirements. Plain English helpers will always alert the user to any issue before appending to the database. Administration is designed to be as simple as possible so the only options are to delete and add using the + and – icons, showing only relevant fields at the time. The system also checks for orphan users that are a member of the group to be deleted and will prevent group deletion until they are deleted. There can be many users to the system and they can be managed from this tab. A simple create, read and delete interface has been provided for user administration. Users can be added if certain validation proves true such as the password containing at least six character is contains both alphabetic and numeric characters.

As with groups, no fields are visible until a request to add a user has been made. This is to simplify the interface to the system administrator. The system was designed to be extensible from the outset, the system administrator can on demand register and de-register libraries as needed. Buggy libraries can also be removed if necessary such as development versions and new versions be loaded on the fly without restarting the whole application. More importantly whilst version 1.2 solely targets text based steganography, version 2 will support other plugins such as encryption and compression by using a prioritisation system (only minimal change in the core application is required to accomplish this).

**Figure 9** Implementation – libraries (see online version for colours)



| | Library ID | Library | Path |
|---|---|---|---|
| 1 | F24A1DB7-CBE4-4415-AD2D... | Mimic++ | /home/bobbyjo/StegAid/buil... |
| 2 | A068BED5-65A0-49A5-97D2-... | US/UK English Bit Represent... | /home/bobbyjo/StegAid/buil... |
| 3 | 6CD6CF4E-4054-4A9A-AA4B-... | Open Space Stego Method | /home/bobbyjo/StegAid/buil... |
| 4 | 50A26A3C-EB26-44F5-83D8-... | Stegonagraphic Synonym Re... | /home/bobbyjo/StegAid/buil... |

To add a library, the + icon is clicked. A 'select file' option is provided that loads a file dialogue box showing only relevant *.dll or *.so files. The library is then loaded and tested to ensure it meets the criteria before registering it to the system. Valid libraries have a unique class identifier and a useful description. If the mail option is enabled, another interface becomes available, the mail administration area. A user can have one email address but many protocols to support the email capability. Further to this there may also be backup servers and so forth. Upon changing the protocol, the default standard port changes as appropriate but is amendable as some servers have irregular ports.

## 4 Evaluation

Each method was evaluated using set criterion (both qualitative and quantitative) and displayed via a result set like shown in Figure 10.

We can see from the example diagram the evaluation is based on five benchmarks:

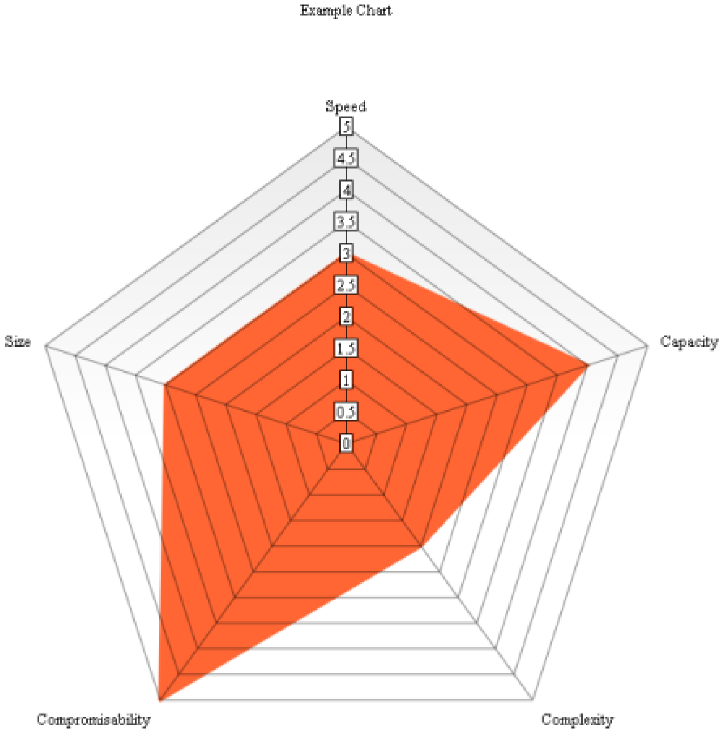
- Speed – The speed in milliseconds for the encoding to take place. Note in the interested of fairness, our versions are all single thread and uncompressed. The more speed the lower the score.
- Capacity – The number of bits that can be encoded within set limit. The less capacity the lower the score.
- Complexity – The difficulty (estimated) in implementing the algorithm. The more complexity the lower the score.
- Compromisability – The difficulty (without knowing the algorithm) on suspecting hidden information. The ease of breaking the algorithm the lower the score.
- Size – The result set size in comparison to other methods. The more the result set size the lower the score.

The target for any algorithm is to have a score of 5 in all key areas:

The application evaluation is the process of applying critique to the resulting software. Evaluation takes multiple forms and includes the following methods:

- Usability testing – The evaluation of responses by ten users who have tried this software in order to obtain directions for either improvement in the current release or a future release.
- Project evaluation – The personal evaluation of the project detailing issues that have arisen, what should be done to improve and recommendations should a future release occur. Time constraints or other reasons will be noted as to why these are not incorporated into this first release.

**Figure 10** Example evaluation radial chart (see online version for colours)



### *4.1 Steganographic methods*

In order for a fair testing and evaluation take place all methods use a single thread (as some can only be performed sequentially). Whilst Stegaid is perfectly capable of compression and encryption (should a plugin be developed), no algorithms shall make use of a these technologies. Some research papers were using compression and/or encryption in their tests and are disregarded in this application.

#### *4.1.1 Open space encoding*

This method is the simplest method that was chosen, which involves the encoding of bits between the spaces of words. Depending on the implementation, the encoding can be different. In this implementation, a simple option was chosen whereby a zero bit (off) has one space and a one bit (on) has two spaces. The encode limit is 1 bit per word and across 1,000 words, therefore 125 bytes of information can be encoded. The current implementation using a randomly generated document from Project Gutenberg, an image of 513 Bytes could be encoded into 5,205 bytes.

| *Encode* | *Partial result* | *Decode* |
|---|---|---|
| | The Project  Gutenberg EBook of Beyond Good and Evil, by Friedrich Nietzsche<br>This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at http://www.gutenberg.org | |

The tests in this implementation show that whilst open space is a simple solution however only small images such as icons and/or a small amount of textual information can be stored.

#### *4.1.2 Synonym replacement*

The process of using a thesaurus which is identical on both clients can be used to store information. For each word in a given text a word is replaced with a synonym of that word. Our implementation is simpler in that a given word is searched through the thesaurus to generate a suitable word list. The greater the number of words in the generated list, the more bits can be encoded (variable bit encoding). If there are three synonyms, two bits can be encoded or seven synonyms 3 bits. In some cases there are more than 60 synonyms for a given word so the number of bits that can be encoded is higher. It is noted that currently this plugin does not identify a context for the given synonym, for example:

> "The 'cat' sat on my lap."
>
> "The 'lion' sat on my lap."

Logically this sentence is correct but does not make sense given the average weight of a lion is very heavy and do not make good lap pets.

**Figure 11** Open space algorithm evaluation (see online version for colours)

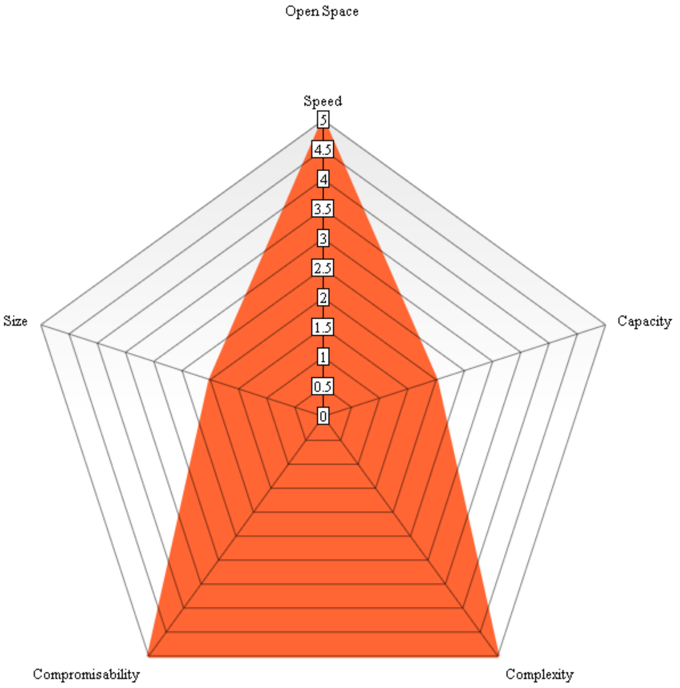
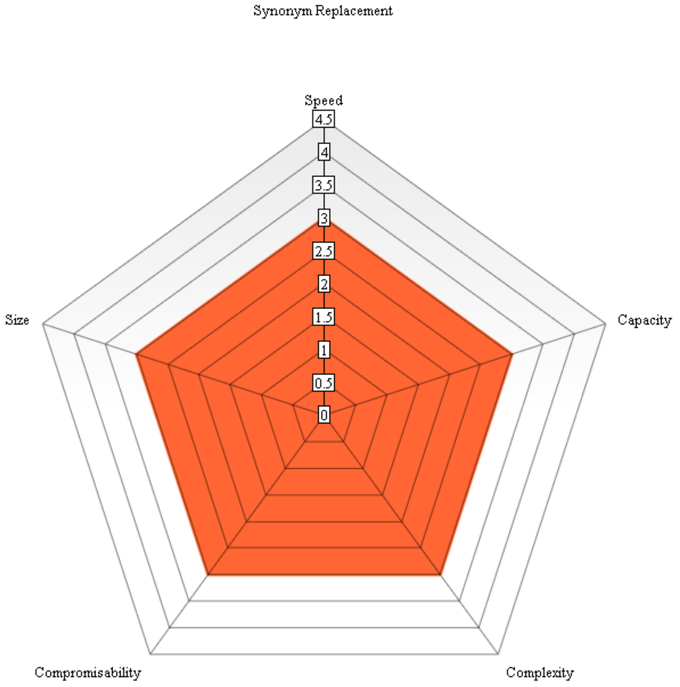


**Figure 12** Synonym replacement algorithm evaluation (see online version for colours)



| *Encode* | *Partial result* | *Decode* |
|---|---|---|
| *Hello World!* | The Beget Gutenberg EBook anent ‘Smiles’, obsolete Eliot H. Robinson<br>This eBook is hereby the benignity atop anybody anywhere helium einsteinium cost borrow with almost europium restrictions what. Her may blowup me, bear alterum backwards dolophine | Hello World! |

The issue with our implementation is that in some cases the sentence makes sense, other parts of the sentence do not. The system has no way of knowing that the grammar makes semantic sense.

### *4.1.3 English diversity*

Given the difference in spelling between UK and US English, these differences in itself can be used as a storage media. Whilst simple to implement an extremely large text could only encode a limited amount of information. The problem with this solution is the limited number of words with differences between the UK English language and the US variant. In our tests a large amount of text is required to encode even a short message, obviously this is entirely dependent on the number of words that can be translated exist. In the current implementation, the UK English variant represents zero and US variant represents one.

| *Encode* | *Partial result* | *Decode* |
|---|---|---|
| Hello World! | If there be any one feature in this textbook more to be commended than another, it is the exposition in Part III. The situations arising in many different kinds of business are here analyzed. | Hello World! |

In our results, we were able to encode only limited information. The encoding of ‘Hello World’ took 146 KB using a Gutenberg text. To maximise the encode rate in future it is therefore suggested that a text be created that specifically has words in the US/UK cross dictionary.

### *4.1.4 Wayner’s mimic functions*

This method tested does not use a pre-existing text to store information, instead using a random selected grammar file; phrases are encoded depending on the bits that are required to be stored. The recipient must have the same grammar file in order to successfully decode the data. Issues arise when the end of the grammar file is reached and the cover text information becomes repeated. The information that can be encoded (as the cover text is generated) is unlimited, however, because of the previous point it is not recommended in case of discovery.

Our implementation differs slightly from the original C code developed by Wayner (1991). Unfortunately the application no longer operates on Microsoft Windows or compiles successfully. Wayner’s version uses a probabilistic approach using a syntax such as: *AAStart = Fred went to *con/.1/.AAStart is the start of the text (the variable). ‘Fred went to’ > Next Variable and the weight is the final number, the higher the weight, the more probable (Defcon, 2003). Our implementation differs, using the same grammar files we can encode bits by counting the appropriate variables and identifying the number of bits we can embed using a counting algorithm. So if there are two values corresponding to a variable, we can encode two bits, 0 or 1. The more values to each variable, the more bits can be encoded.

**Figure 13** UK/US English algorithm evaluation (see online version for colours)

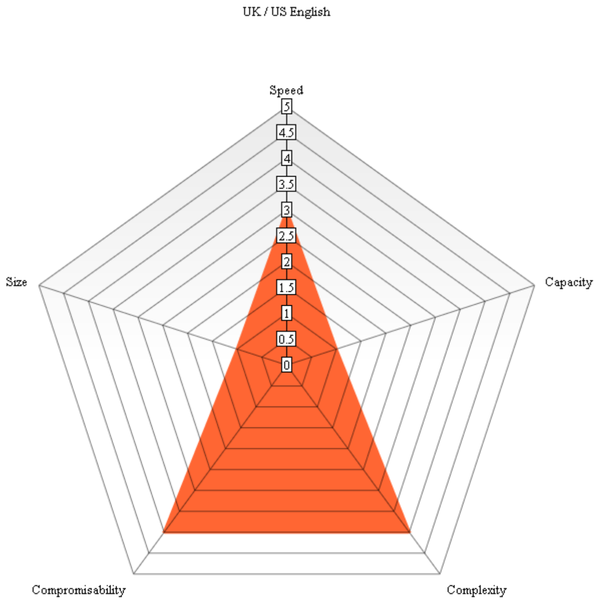
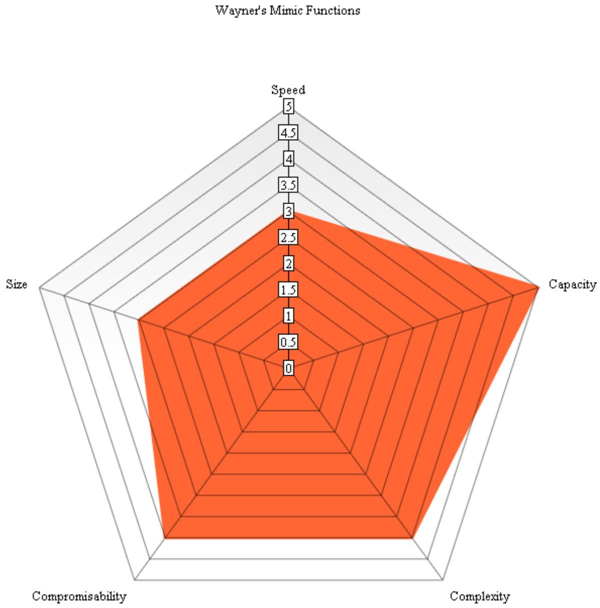


**Figure 14** Wayner's mimic functions evaluation (see online version for colours)



| *Encode* | *Partial result* | *Decode* |
|---|---|---|
| Hello World! | Let's get going! Top of the inning. | Hello World! |

Through this evaluation we have tested and evaluated four Steganographic algorithms that can be used to hide information such as files and documents. A summary is presented in Table 1.

**Table 1** Evaluation summary

| *Method* | *Summary* |
|---|---|
| Open space encoding | Open space encoding was the simplest method to implement. Through our implementation it was shown that capacity is approximately 10% of the size of the cover text (language dependent). |
| Synonym replacement | Synonym replacement (the process of changing words with comparable thesaurus words). The better the thesaurus, the greater the probability of occurrence of a synonym. Our specific implementation checks for a word and replaces it using a thesaurus. It was noted, however, that the current library is unable to discern the context of the grammar at hand. For example, project (hurl) and project (task) are the same words but have different meanings. Test results indicate that Synonym replacement has a higher encode rate than that of Open Space encoding. |
| UK/US English translation algorithm | This method involves the use of encode bits (zeroes and ones) by way of selecting UK (or US) English as bit zero and the other as bit one. Whilst in itself would be unexpected to a reader of the cover text to notice the difference, the capacity is extremely small. This is due to the likelihood in occurrence of a word that is spelled differently across the two nations. |
| Wayner's mimic functions | Wayner (1991) developed the mimic functions to mimic natural language using a Grammar File to generate context free grammar. It was found that the encode rate was low, the more variable paths in a grammar file the more information could be encoded. Whilst having a low bitrate, our versions of the Mimic Functions have an unlimited capacity. Once the end of a path is reached, the beginning of the path is taken again. This would cause repeating text which would be suspicious to users viewing the cover text. |

## 5 Conclusions

Various algorithms with relation to text based steganography have been investigated, four of which have also been implemented. This research firstly involved the investigation of various methods within steganography as a whole and drilled down further specifically into text based steganography. The next step involved the investigation into Requirements of a potential system in order to provide a direction for the project to move forward. Whilst most steganographic research focuses on a single method of encoding such as the famous mimic functions, (Wayner, 1991), this implementation focuses on four, although not limited to four. The eventual design was designed with future enhancements in mind. To aid further in future development, the current user interface (the view) can be 'stripped' and replaced without affecting how the application operates. In fact the application can support multiple Views, as the logic within the application does not reside here. The application logic and validation occur within the base classes (controllers) and the data is provided within the embedded database system. To enable further future proofing the use of a plugin architecture is provided, thus allowing an administrator to register new methods as they become available. Whilst in this version we concentrated solely on text based steganography, the system in its current form can be used with other steganography methods, encryption or compression out of the box.

The application designed is not inherently aware of the encoding of the cover text. To date all known steganographic applications encode only one method, it is obvious to that application as to which method must be used. Stegaid uses multiple methods of encoding depending on the user's choice. As such the information (binary bits) are encoded but not how it is encoded. The receiver must know how the method used in order to decode the information. In future, the application could encode information about the encoding also, which raises another issue. If the embed how the information is encoded into the cover text, any peer network users can instantly decode the text (if they know the information is suspicious).

Finally, Stegaid is not limited to steganography but could potentially perform encryption and compression as part of a hybrid system. This would make Stegaid far more secure in that data is hidden but if discovered, still encrypted. Given recent developments with the Heartbleed bug in OpenSSL (Jeske et al., 2017; Gupta and Gupta, 2016; Harran et al., 2017) it makes sense to have an additional level of security beyond encryption.

## References

Agarwal, M. (2013) 'Text steganographic approaches: a comparison', *International Journal of Network Security & Its Applications*, Vol. 5, No. 1, pp.91–106.

Al-Nabhani, Y. et al. (2010) 'A new system for hidden data within header space for EXE-File using object oriented technique', *Kuala Lumpur, 3rd IEEE International Conference on Computer Science and Information Technology* (*ICCSIT*), pp.9–13.

Asad, M., Gilani, J. and Khalid, A. (2001) 'An enhanced least significant bit modification technique for audio steganography', in *2011 International Conference on Computer Networks and Information Technology* (*ICCNIT*), Rawalpindi.

Atawneh, S., Al Bazar, H., Usman, I and Sumari, P. (2016) 'Secure and imperceptible digital image steganographic algorithm based on diamond encoding in DWT domain', *Multimedia Tools and Applications*, pp.1–22, 10.1007/s11042-016-3930-0.

Balaji, R. and Naveen, G. (2011) 'Secure data transmission using video steganography', *IEEE International Conference on Electro/Information Technology* (*EIT*), Chennai, pp.1–5.

Barilnik, S.S., Minin, I.V. and Minin, O.V. (2007) 'Adaptation of text steganographic algorithms for HTML', in *8th Siberian Russian Workshop and Tutorial on Electron Devices and Materials*, Siberia.

Bennett, K. (2004) *Linguistic Steganography: Survey, Analysis and Robustness Concerns for Hiding Information in Text*, Center for Education and Research in Information Assurance and Security, West Lafayette.

Bhattacharyya, S., Banerjee, I. and Sanyal, G. (2010) 'A novel approach of secure text based steganography model using word mapping method (WMM)', *International Journal of Computer and Information Engineering*, Vol. 4, No. 2, pp.96–103.

Chang, C-Y. and Clark, S. (2010) 'Linguistic steganography using automatically generated paraphrases', *Proceedings of the Annual Meeting of the North American Association for Computational Linguistics*, Los Angeles, pp.591–599.

Cheddad, A., Condell, J., Curran, K. and McKevitt, P. (2010) 'Digital image steganography: survey and analysis of current methods', *Signal Processing*, Vol. 20, No. 3, pp.727–752.

Cuttnel, J.D. and Johnson, K.W. (1998) *Physics*, 4th ed., Wiley, New York.

Gopalan, K. and Wenndt, S. (1998) *Audio Steganography for Covert Data Transmission by Imperceptible Tone Insertion* [online] http://www.purduecal.edu/engr/docs/GopalanKali_422_049.pdf (accessed 1 February 2017).

Gupta, B.B., Agrawal, D. and Yamaguchi, S. (2016) *Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security*, IGI Global, New York, USA, ISBN-13: 978-1522501053.

Gupta, S. and Gupta, B.B. (2016) 'XSS-secure as a service for the platforms of online social network-based multimedia web applications in cloud', *Multimedia Tools and Applications*, pp.1–33, https://doi.org/10.1007/s11042-016-3735-1.

Harran, M., Farrelly, W. and Curran, K. (2017) 'A method for verifying integrity & authenticating digital media', *Applied Computing and Informatics*, Vol. 13, No. 2, pp. in pre-print, DOI: 10.1016/j.aci.2017.05.006, ISSN: 2210-8327, Elsevier

Hernon Moraldo, H. (2012) 'An approach for text steganography based on Markov chains', *4th WSEAS Workshop on Computer Security*.

Jalil, Z. and Mirza, M.A. (2009) 'A review of digital watermarking techniques for text documents', *International Conference on Information and Multimedia Technology*, pp.230–234.

Jeske, D., McNeill, A., Coventry, L. and Briggs, P. (2017) 'Security information sharing via Twitter: 'Heartbleed' as a case study', *International Journal of Web Based Communities*, Vol. 13, No. 2, pp.172–192

Li, J., Yu, C., Gupta, B.B. and Ren, X. (2017) 'Color image watermarking scheme based on quaternion Hadamard transform and Schur decomposition', *Multimedia Tools and Applications*, pp.1–17, DOI: 10.1007/s11042-017-4452-0.

McBrearty, S., Farrelly, W. and Curran, K. (2017) 'The performance cost of preserving data/query privacy using searchable symmetric encryption', *Security & Communication Networks*, Vol. 9, No. 18, pp.5311–5332, DOI: 10.1002/sec.1699.

Rig, D. and Tuithung, T. (2012) 'A novel steganography method for image based on huffman encoding', *3rd National Conference on Emerging Trends and Applications in Computer Science* (*NCETACS*), Vol. 14, No. 18, pp.30–31.

Shirali-Shahreza, M. (2008) 'Text steganography by changing words spelling', *10th International Conference on Advanced Communication Technology*.

Thinkmap Inc. (2013) *Visual Thesaurus – Relationships* [online] http://www.visualthesaurus.com/howitworks/manual/#reldesc (accessed 1 February 2017).

Topkara, U., Topkara, M. and Atallah, M.J. (2006) 'The hiding virtues of ambiguity: Quantifiably resilient watermarking of natural language text through synonym substitutions', *Proceedings of the 8th Workshop on Multimedia and Security*, New York.

Wayner, P. (1991) *Mimic Functions – The Manual* [online] http://ftp.ccu.edu.tw/pub/crypt/old/mimic/Mimic-Manual.txt (accessed 12 November 2016).

Yu, C., Li, J., Li, X., Ren, X. and Gupta, B.B. (2017) 'Four-image encryption scheme based on quaternion Fresnel transform, chaos and computer-generated hologram', *Multimedia Tools and Applications*, pp.1–24, DOI: 10.1007/s11042-017-4637-6.