I.J. Mathematical Sciences and Computing, **2019, 1, 18-30**
Published Online January 2019 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijmsc.2019.01.02

Available online at ***http://www.mecs-press.net/ijmsc***

Low-Tech Steganography for Covert Operations

Akash Nag[*]

Department of Computer Science, M.U.C. Women's College, Burdwan 713104, WB, India

Received: 21 April 2018; Accepted: 09 August 2018; Published: 08 January 2019

Abstract

Text steganography, the art of concealing a secret text inside another innocuous text called the cover, is usually performed by insertion of whitespace, punctuation marks, misspelling words, or by arbitrarily capitalizing words or inserting synonyms, changing font-sizes & colors, etc. All of these have the disadvantage that they either arouse suspicion or are easily noticeable; and even lost if manually copied, i.e. handwritten. Furthermore, they are easily detectable by automated checkers. Still there are other methods which require a stego-key in order to decrypt the message. In covert intelligence operations, transmission of the stego-key may not be possible at all, more so when the message is urgent. Digital communications and Internet connectivity may also be lacking in certain situations, and the only mode of message passing available may be the exchange of handwritten text on paper; which effectively rules out text modifications like font-changes, whitespace insertion, etc. or any form of digital steganography like image/audio steganography. Finally, in almost all text-steganographic techniques, there is no provision to for the receiver to detect whether or not there is indeed any message embedded. This is very important in intelligence operations where a number of decoy text need to be sent with only one concealing the actual message. In this paper, we propose a new tool called STEGASSIST that can help the sender in generating the stego-text manually. It is a low-tech form of steganography that is especially suited to covert operations like espionage or under-cover journalism. In this method, the generated cover and the stego-text are identical, or in other words, there is no cover-text. Moreover, decryption does not require a stego-key, and the stego-text may be printed or even hand-written and sent via unreliable messengers, or published, without arousing any suspicion. Finally, the received stego-text can be checked by the receiver to detect whether or not there is any actual message embedded in it.

Index Terms: Low-Tech Steganography, Text Steganography, cover generation, espionage, under-cover journalism.

* Corresponding author.
E-mail address: psrgietcse@gmail.com

1. Introduction

During the first half of 2000, it was reported by Jack Kelley in *USA Today* that terrorists were using steganography to conceal messages in pornographic images in order to communicate with one another [1]. It created an overnight sensation and prompted intelligence agencies and researchers alike to develop new tools that can detect concealed messages. However, as time passed, it gradually became clear that terrorist organizations were more interested in low-tech steganography than digital steganography. This is because, low-tech steganography was far easier to implement, and far more difficult to detect by intelligence agencies than digital steganography against which a number of statistical detection techniques have been developed. However, media outlets continued to report otherwise, and still many people today believe that terrorists use digital steganography instead, although no concrete evidence exists for the same. The use of cryptography by terrorists however is a substantiated claim, as many encrypted files have been found on computers belonging to terrorists [2, 3].

What drives terrorists to use low-tech steganography is reflected in the ground reality of the environments in which they operate. The lack of uninterrupted digital communications, and the increasing reach & computational resources of intelligence agencies make the use of digital steganography risky and unreliable. A similar parallel can be drawn to agents who are on a covert intelligence gathering operation in a hostile or allied country. Since every intelligence agency around the world spies on every other agency, digital steganography is hardly a trustworthy medium of exchanging sensitive intelligence. Moreover, in war-torn areas or critical situations, digital communication channels may not exist at all, making digital steganography useless and impractical. For example, the local government may totally shut down the Internet in the area in order to prevent leakage of sensitive information during illegal operations like mass genocide. In such situations, the only medium of information exchange may be through handwritten notes passed from one person to another, possibly via several unreliable intermediaries. Cryptography cannot be used in most cases, not only because of the lack of communication channels but also because of the fact that even the suspicion that secret messages are being sent may be fatal for the sender. The only form of secret communication that can be used then is low-tech steganography. Low-tech steganography is a technique that can be used to hide messages using non-digital means, e.g. using paper and pencil, a message can be constructed that can be handwritten and sent via a courier/messenger. In this paper, we propose a new low-tech steganographic approach that is specially suited for covert operations like espionage or under-cover journalism, in the absence of digital communication channels. It has the advantage that messages can be embedded in any text, be it newspapers, invoices, books or research articles, and hence it is impossible to check each and every piece of text everywhere. Even if one becomes suspicious, low-tech steganography may be impossible to prove, because the stego-text may just be an innocuous text after all, and any secret message that is extracted from it may just be a coincidence.

2. Related Work

Most research in the domain of steganography has been focused on digital steganography, primarily image and audio steganography. But as explained before, digital steganography has its limitations in practice. Recently, a number of text steganographic techniques have been developed, which include whitespace insertion [4, 5, 6], punctuation insertions [7], article insertions [8], misspelling words [9], font color changes [10], line-shift and word-shift coding [11], dot-spacing [12], grammar-based cover generation such as the NICETEXT program [13], inserting synonyms [14], etc. Some other methods include sending a cheating index-file along with the stego text-file to help the receiver to reconstruct the message [15, 16], but the problem with these approaches is that if the cheating file is intercepted, the attacker can decode the message. According to Bennett [17], all of these methods can be categorized into three groups: format-based, statistical, and linguistic methods. While some format-based methods may be visible to the human-eye, many are not. But all of them however can be detected using software. To bypass automatic filtering by software, statistical and linguistic methods were developed, but they will not pass checking by humans because the generated text will be mostly nonsense, although the sentences being syntactically correct and meaningful individually. Most of the text-based steganographic techniques named above require a computer to generate and must consist of printed text or text sent digitally. They cannot be used for handwritten communication. Even those techniques which can be applied to handwritten text, such as misspelling words deliberately, will raise suspicion if intercepted. Another form of steganography is the use of microdots [18], where text or even entire images are compressed and encoded in dots. However, this also requires computers to generate and/or decode them.

Not much structured research has been undertaken in the field of low-tech steganography. The most common low-tech steganographic technique is the use of null ciphers where every *K*-th letter is extracted to reconstruct the secret message [19]. A prominent example of this technique can be found in World War II, when a Nazi spy sent the following text:

```
Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade
   issue affects pretext for embargo on by-products, ejecting suets and vegetable oils.
```

If we extract the 2nd letter of every word, we are left with the original message which is:

```
Pershing sails from NY June 1.
```

Null ciphers are easy to break once it is suspected that they are being used. What is required is a strong yet low-tech steganographic approach that can withstand detection and decoding. We now describe a novel method of low-tech text-steganography in Section 3, and compare it against both digital and text steganographic techniques in Section 4.

3. The Proposed Algorithm

The algorithm proposed in this paper, hereinafter called STEGASSIST, helps the sender in generating the stego-text, for a given secret-text. The process works only for text inputs. The program is not a stego-text generation tool per se, but is instead a type of checker. When given a stego-text and a secret-message as inputs, the program checks to see whether the stego-text can indeed be used to conceal the message. If not, it suggests additions and alterations so that the concealment can be performed. Unlike other steganographic tools, a cover-text is not required as the stego-text may be thought of being the cover-text as well.

The program works in two modes which are synonymous with insertion and extraction of secret messages respectively:

1. Stego Text Generation Mode (Insertion)
2. Secret Text Extraction Mode (Extraction)

In the rest of this section, we will first outline the working of STEGASSIST in the insertion and extraction modes, after which we will elaborate on the various phases of the insertion procedure in details.

3.1. Stego Text Generation

In this mode, a stego-text is generated by the user manually while STEGASSIST assists the user in doing so. The stego-text generation process is illustrated in Fig. 1. The stego-text is generated in two phases:

1. **Word embedding:** First, the user inputs the secret-text and a cover-text, while STEGASSIST checks to see if the secret-text can be effectively concealed within the cover-text. This phase can work in either of two modes: whole-word matching, or partial-word matching. If the cover-text can be used, STEGASSIST generates a textual code, called the Position Vector Code (PVC) as the output of the first phase. During PVC generation, the user may ask for the data to be optionally encrypted and a hash

appended at the end.

2. **Position vector embedding:** In the second phase, STEGASSIST asks the user to append additional sentences to the cover-text so that the PVC can be embedded. Specifically, it asks the user to include such words in the additional sentences which start with letters contained in the PVC.

3.1.1. Prerequisites for the Cover Text

Since the stego-text and the cover are identical, and there is no stego-key, the cover-text must then be of some particular structure in order to hide data in it. The basic prerequisites for the cover and secret text are:

1. The cover must contain all the words of the secret message, either in whole, or in part in the form of a prefix or a suffix of any word.
2. Every word v, of a pair of consecutive words u and v in the secret message must not be placed more than $2N$ words apart from the word u; where N is an input parameter.
3. Proper nouns beginning with a capital letter (other than "I"), except those specifically asked for by STEGASSIST, must be present only at the beginning of any sentence in the cover.

3.1.2. Input Parameters

The STEGASSIST program takes several user-parameters, which are listed in Table 1. As can be seen, the stego/cover text must also be given as input, and the program only checks the validity of the text for steganographic concealment. If it is invalid, STEGASSIST suggests changes that need to be made to the text.

3.1.3. Position Vector Generation

The process of generating a position vector begins by verifying whether each word of the secret message appears in the cover text, either wholly or partially, depending on whether the user has allowed partial matches. If partial matches are prohibited, the process can be sped up using a hash-table. The list of words in the cover text is first put into a hash-table; and then each word of the secret message is hashed to determine whether or not it exists in the cover. If partial matches are allowed, the secret word must be present either as a prefix or as a suffix in any word of the cover. Substring matches in the middle of another word are not allowed. The second step consists of the actual encoding, which varies according to whether partial matches are allowed.

Table 1. Parameters during Encoding

Parameter	**Range of accepted values**	**Required / Optional**	**Default**	**Meaning**
Secret-text	Any English text without special characters	Required	-	The secret message that would be concealed
Cover/Secret Text	Any paragraph of suitable English text	Required	-	The innocuous text in which the message would be concealed
N	$N=2^k$ for some integer k, where $N \geq 8$	Optional	32	Half of the max. difference (in no. of words) between the positions of 2 consecutive words of the secret message in the cover
Encrypt	Yes/No	Optional	No	Whether to encrypt the position vector using Blowfish
Encryption Key	32-448 bits	Optional	-	The key to use during encryption
Allow partial match	Yes/No	Optional	No	Whether to allow partial word matching
Append hash	Yes/No	Optional	No	Whether to allow appending the hash of the position vector for error checking at the receiver end

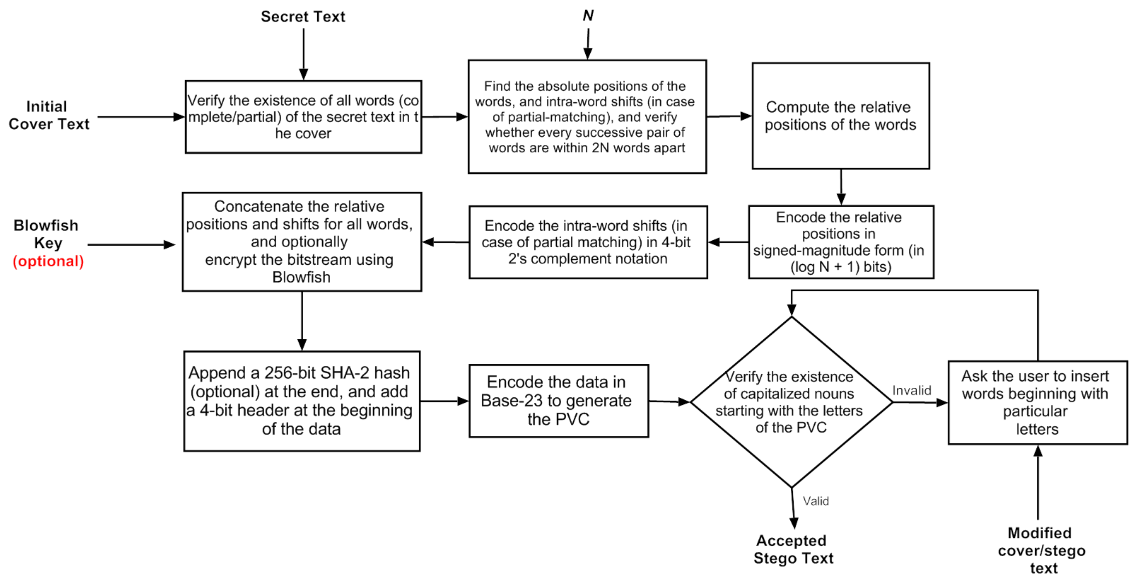
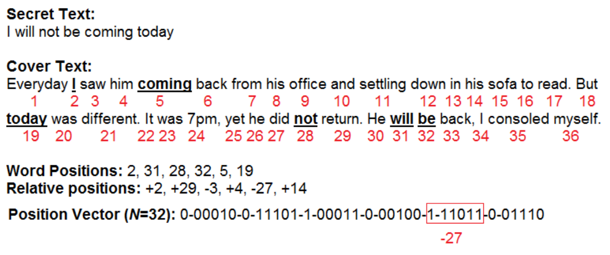

Secret Text
N
Initial Cover Text
Verify the existence of all words (complete/partial) of the secret text in the cover
Find the absolute positions of the words, and intra-word shifts (in case of partial-matching), and verify whether every successive pair of words are within 2N words apart
Compute the relative positions of the words
Blowfish Key (optional)
Concatenate the relative positions and shifts for all words, and optionally encrypt the bitstream using Blowfish
Encode the intra-word shifts (in case of partial matching) in 4-bit 2's complement notation
Encode the relative positions in signed-magnitude form (in (log N + 1) bits)
Append a 256-bit SHA-2 hash (optional) at the end, and add a 4-bit header at the beginning of the data
Encode the data in Base-23 to generate the PVC
Verify the existence of capitalized nouns starting with the letters of the PVC
Invalid
Ask the user to insert words beginning with particular letters
Valid
Accepted Stego Text
Modified cover/stego text

Fig.1. The Proposed Method For Stego-Text Generation

3.1.3.1. *Whole Word Matching:*

If partial matches are disabled by the user, the first occurrence of the first secret word in the cover is found. Its position, say P, in terms of number of words, is determined and recorded in M-bit binary prefixed by a zero, where $M = \log_2 N$. For the next secret word, positions $P–N$ to $P+N$ are searched in the cover to find a match. If no match is found, the program aborts the encoding process. If the next word is found, say at position Q, the value of $Q–P$ is recorded in (M+1)-bit binary in signed-magnitude form, with the MSB being the sign-bit. This process continues till the positions of all words have been encoded. Since for each word in the secret text, a (M+1)-bit position vector is generated, the length of the position vector is therefore $W(\log_2 N + 1)$ bits in length, with W being the number of words in the secret message. In Fig. 2, we give an example to clarify this process of encoding.


Secret Text:
I will not be coming today

Cover Text:
Everyday **I** saw him **coming** back from his office and settling down in his sofa to read. But
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
today was different. It was 7pm, yet he did **not** return. He **will** **be** back, I consoled myself.
19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36

Word Positions: 2, 31, 28, 32, 5, 19
Relative positions: +2, +29, -3, +4, -27, +14
Position Vector (*N*=32): 0-00010-0-11101-1-00011-0-00100-$\boxed{\text{1-11011}}$-0-01110
-27

Fig.2. Position Vector Generation if Partial Matches Are Disabled

3.1.3.2. *Partial Word Matching:*

If partial matches are allowed, the encoding process is similar, except that each position vector is now appended with a 4-bit offset. This offset stores the position of the prefix/suffix relative to the selected cover word, in 2's complement notation. If it is a prefix, the range is 0–7 shift in characters. If it is a suffix, it is encoded as a negative number in the range of –8 to –1. The length of the position vector in this case is therefore $W(\log_2 N + 5)$ bits in length. The limitation with the whole-word match scheme described before is that sensitive words of the secret message, when found in the stego-text, will raise suspicion. In this scheme of partial matching, sensitive words of the secret-text may be broken up by the sender manually or even successive words may be combined, and then partially matched as prefix/suffix with innocuous words in the cover. The cover-word must however either start or end with the secret-word, and cannot contain the latter in the middle. The offset or intra-word shift calculation is clarified with an example in Fig. 3(a). The encoding process, with partial-matching allowed, is illustrated in Fig. 3(b) with an example.

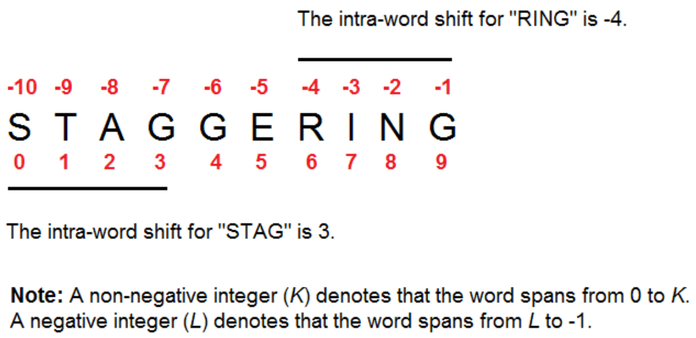
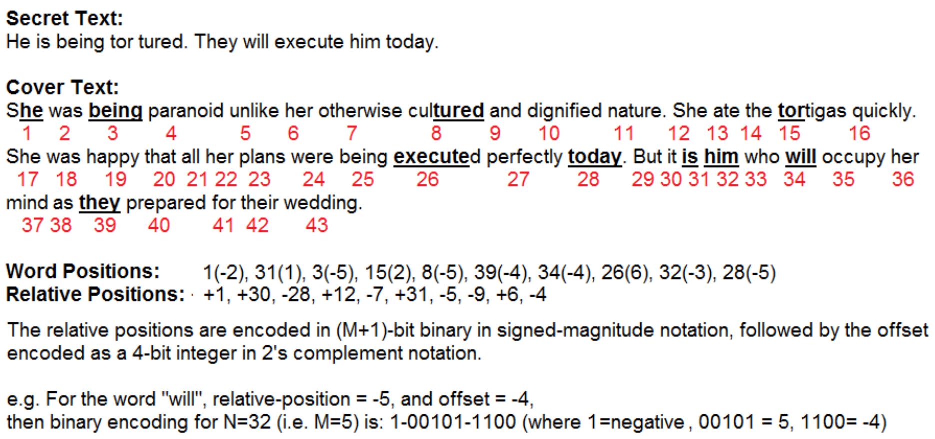

The intra-word shift for "RING" is -4.
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1
S T A G G E R I N G
0 1 2 3 4 5 6 7 8 9
The intra-word shift for "STAG" is 3.
Note: A non-negative integer (K) denotes that the word spans from 0 to K.
A negative integer (L) denotes that the word spans from L to -1.

Fig.3(a). The Offset or Intra-Word Shift Calculation



Secret Text:
He is being tor tured. They will execute him today.

Cover Text:
S**he** was **being** paranoid unlike her otherwise cul**tured** and dignified nature. She ate the **tor**tigas quickly.
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
She was happy that all her plans were being **execute**d perfectly **today**. But it **is** **him** who **will** occupy her
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
mind as **they** prepared for their wedding.
37 38 39 40 41 42 43

Word Positions: 1(-2), 31(1), 3(-5), 15(2), 8(-5), 39(-4), 34(-4), 26(6), 32(-3), 28(-5)
Relative Positions: +1, +30, -28, +12, -7, +31, -5, -9, +6, -4

The relative positions are encoded in (M+1)-bit binary in signed-magnitude notation, followed by the offset encoded as a 4-bit integer in 2's complement notation.

e.g. For the word "will", relative-position = -5, and offset = -4,
then binary encoding for N=32 (i.e. M=5) is: 1-00101-1100 (where 1=negative , 00101 = 5, 1100= -4)

Fig.3(b). Position Vector Generation if Partial Matches are allowed

3.1.4. Header and Trailer Generation

The position vector thus generated using either of the two schemes described above, i.e. whole-word match, or partial-word match, is then optionally encrypted using the Blowfish algorithm [20] if the user has specified the key. The encrypted or unencrypted position vector is then prefixed by a 4-bit header as shown in Fig. 4. The header starts with a 1 so as to detect where the header starts during the secret-text extraction phase. Although the header is mandatorily included, the trailer is optional. If included, it consists of a 256-bit hash. The position-vector, appended with the value of N (as a 6-bit integer) is hashed using the SHA-2 algorithm, and appended to the end of the packet as shown in Fig. 5. Note that the value of *N* is never actually stored in the packet.

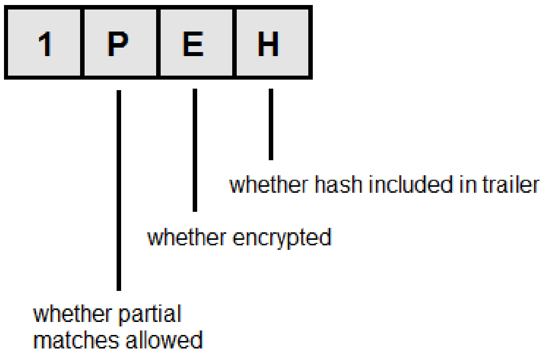

1
P
E
H
whether hash included in trailer
whether encrypted
whether partial matches allowed

Fig.4. The 4-bit Mandatory Header

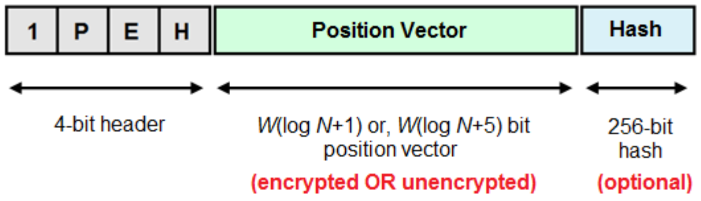
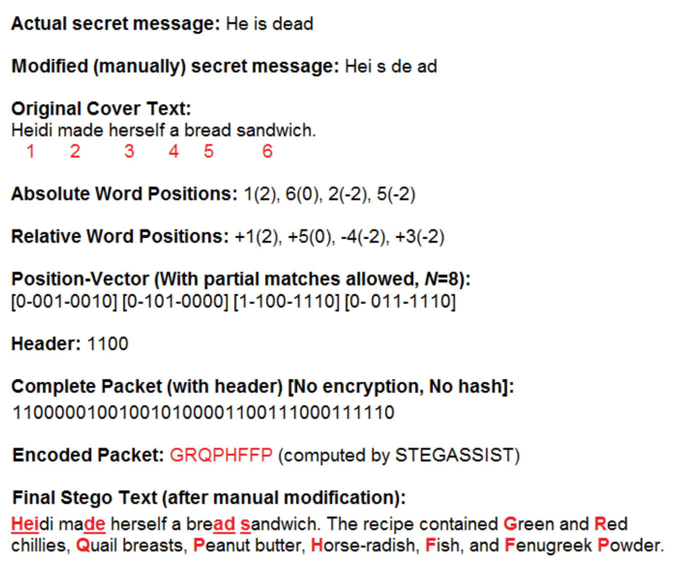

1
P
E
H
Position Vector
Hash
4-bit header
W(log N+1) or, W(log N+5) bit position vector
(encrypted OR unencrypted)
256-bit hash
(optional)

Fig.5. The Complete Packet Format

3.1.5. PVC Generation and Noun Capitalization

The complete packet obtained in the previous step is then encoded in base 23, with the letters A–W substituted as the digits, with A being 0 and W representing 22. This encoded vector, called the PVC (Position Vector Code) is then concealed in the cover using capitalization. The cover-text is searched from left to right for proper-nouns starting with the letters in the encoded vector. Nouns which are neither “I”, nor occurring in the beginning of a sentence, are selected for this purpose. For each such noun, the first letter is capitalized, as it should be in English – thus not raising any suspicion. The proper nouns must be present in the same order as the letters in the encoded vector. Care must be taken to ensure that no other proper nouns exist in the cover as they would be in lowercase, thus raising suspicion; since in English all proper nouns must begin with a capital letter. In essence, it works as a null cipher. This encoding process is largely manual, as the sender is in a better position to judge which nouns should be capitalized and which not. After each modification, the STEGASSIST program may be run interactively which would inform the user about what capital character is needed next so that the user can better decide what words to include. The complete encoding process is illustrated in Fig. 6. The generated PVC in the example is GRQPHFFP, which is then embedded using noun-capitalization.



Actual secret message: He is dead

Modified (manually) secret message: Hei s de ad

Original Cover Text:
Heidi made herself a bread sandwich.
1 2 3 4 5 6

Absolute Word Positions: 1(2), 6(0), 2(-2), 5(-2)

Relative Word Positions: +1(2), +5(0), -4(-2), +3(-2)

Position-Vector (With partial matches allowed, *N*=8):
[0-001-0010] [0-101-0000] [1-100-1110] [0- 011-1110]

Header: 1100

Complete Packet (with header) [No encryption, No hash]:
11000001001001010000110011100011110

Encoded Packet: GRQPHFFP (computed by STEGASSIST)

Final Stego Text (after manual modification):
Heidi made herself a bread sandwich. The recipe contained Green and Red chillies, Quail breasts, Peanut butter, Horse-radish, Fish, and Fenugreek Powder.

Fig.6. Complete Encoding Process with Partial Matching Enabled

3.2. Secret Text Extraction

The secret-text extraction process consists of the following steps:

1. The stego-text must be input by the user, and optionally the value of *N*.
2. STEGASSIST searches for capitalized words, except “I” and words which begin a sentence. It concatenates the first letters of those words to form the PVC, which is then converted to binary from base 23.
3. Any trailing zeroes are discarded, and the first 4-bits are treated as the header. The first bit is always a 1, while the remaining 3 bits are decoded to determine whether encryption, partial-matching or hashing has been used. If encryption has been employed, the program asks the user for the decryption key.
4. If hashing has been used, the last 256 bits of the data is treated to be the hash. The position vector is the data contained between the header and the hash.
5. If the value of *N* has been provided by the user, the position-vector concatenated with the value of *N* is hashed and matched against the extracted hash to verify the message integrity. If no value of *N* has been provided, different values of *N* are iteratively checked in increments of powers of 2, starting from 8.
6. If verification succeeds, the correct value of *N* is determined. If encryption had been used, the data is then decrypted using the key supplied by the user. If verification fails, it means that there is no concealed message or it has been destroyed.
7. The length of the data must be a multiple of $5+\log_2 N$ bits if partial matching has been used; or it must be a multiple of $1+\log_2 N$ bits if partial matching has not been used. If the length of the data is not a multiple of either, there is no concealed message or the message has been tampered with. The number of words in the secret text is determined by dividing the length of the data by the number (among the two choices) of which it is a multiple. The data is then divided into groups, where the group length is either $5+\log_2 N$ or $1+\log_2 N$ bits.

For each group, the first $1+\log_2 N$ bits are decoded from signed magnitude form to decimal to determine the relative position of the word, while the last 4 bits (if present, i.e. if partial matching has been used) is decoded from 2's complement notation to decimal to determine the intra-word shift. These two pieces of information are used to reconstruct the secret message.

4. Results and Discussion

Since the proposed method is largely manual and low-tech, it is not possible to test the algorithm on large textual datasets and report the results. From the very few examples that have been shown above, it appears that the encoding process is robust and resistant to attack. Another problem with measuring the performance of our algorithm is that standard measures of similarity like the Jaro-Winkler distance [21], or histogram comparison do not apply here. This is because all similarity metrics would indicate a 100% similarity between the cover and the stego-text since the stego-text is manually generated by the sender. As far as capacity is concerned, its calculation is also problematic. This is because the generated stego must make sense semantically in order to escape detection. To make it seem innocuous, the sender must lengthen the message by inserting irrelevant but syntactically and semantically meaningful text, which would in turn decrease the capacity. In theory, if no hashing or encryption is used and partial matching is disabled, a *K*-word secret-text requires at least $\left\lceil \frac{K(1+\log_2 N)}{4} \right\rceil + (K+1)$ words in the generated stego-text. However, this lower bound also means that the secret message must appear as-is in the stego-text as well, which of course defeats the entire purpose of steganography. Hence, the actual number of words in the stego-text will be much larger than the lower bound presented here. Since the motivation behind our proposed algorithm is to design a new low-tech steganographic tool that can help in covert operations like industrial or military espionage, undetectability is far more important than performance in these situations.

4.1. Advantages

There are several advantages to be gained with using our proposed algorithm, such as:

1. It can be simulated by hand using only basic equipment like pencil, paper and a calculator if hashing and encryption are not used.
2. It does not require a digital communication channel or medium between the communicating parties.
3. It does not require the exchange of any stego-key.
4. The generated stego/cover text is meaningful and is both syntactically as well as semantically correct.
5. The cover and the stego-text are identical.
6. The stego text can be manually copied by hand or typed on paper, and still retain the secret message. This allows the use of unsuspecting messengers to deliver or transport the message.
7. The output is plain-text and therefore may be embedded in a wide variety of mediums like books, research articles, etc. to conceal it. On the other hand, images may not be that easily concealed.
8. If hashing is used, it can be used to verify both the existence and the integrity of the secret message by the receiver.
9. If encryption and partial matching are used, the security of the scheme is highly enhanced.

4.2. Comparison with other Steganographic Approaches

We will now compare our algorithm with existing steganographic approaches and explain why our approach is superior:

1. **Image and audio steganography:** In image steganography, digital images are exchanged between the sender and receiver, the pixels of which embed the secret data (converted to a bitstream). The most common technique is to use the LSB of each pixel value, or even of each color channel, to embed one bit of the secret message. The advantage of our method over this technique is that this technique presumes the availability of a digital communication channel or medium (maybe a flash-drive), to exchange the image. This may not be possible in war-torn areas where digital communication is lacking or messengers cannot be trusted with digital equipment like flash-drives. Also, innocuous images, when exchanged without any apparent reason, raise suspicions [1]. Audio steganography involves hiding messages in audio, but suffers from the same problem that it requires specialized software/digital equipment to insert and retrieve the secret messages.
2. **Text steganography:** In text steganography, the cover and the stego data are both plaintext. In a sense, text steganography is more complicated than image steganography because it must not only pass through the filters of automated checkers, but also must pass through human checkers without raising alarm. Most text steganographic techniques fall in any one of the following categories:

   (a) **Format-based methods:** Here, the characters of the stego-text are either altered or, characters (including whitespaces) and punctuation marks are inserted or removed. However, this method cannot be used for espionage as too many misspellings, random capitalizations, and unnecessary punctuations & whitespaces will definitely raise suspicion. Moreover, when manually copied, extra whitespaces & punctuations will not be retained, and misspellings may be corrected by messengers. Font-size changes are also usually used but they require word processors, as normal text editors cannot display different font-sizes in the same text. Also, it will be lost when hand-copied.

   (b) **Linguistic methods:** Here, the stego-text is generated either from random dictionary words or from context-free grammars by preserving the syntactic structure of the language. For example, most English sentences have the subject-verb-object structure, and as such, different sentences employing this structure can be generated employing various subject-verbs, object-verbs, etc. However, the resulting sentences are mostly random and although each sentence is meaningful, the text on the whole does not make any sense whatsoever. This method will generate more suspicious stego-text than the format-based methods as intermediaries will not find any reason why a piece of paragraph that does not make any sense, will need to be sent.

   (c) **Null ciphers:** Null ciphering is a manual method involving hiding characters of the secret text in the K letter of every word, or in the every K-th letter of the text. The value of K is the key that needs to be decided between the sender and receiver beforehand. This method, although useful, lacks security because of its regular pattern, and is easily detectable by software through looping over all possible values of K.

3. **Other forms of steganography:** There are numerous other methods of steganography, like hiding in disk-space [22], hiding in network packets [23, 24], hiding in circuitry [25, 26], etc. However, all of these may be highly novel and intelligent ways of concealing data but are practically useless in ground-situations that occur in military espionage, where more often no form of digital communication is available.

4.3. Resistance to Detection and Attacks

The proposed method is highly resistant to detection because of its low-tech nature. The primary objective behind our development of this scheme is so that it can be used on the ground in hand-to-hand communication or exchange of typed text, rather than being digitally sent over a network. Firstly, since the messages are not meant to be sent over a network in digital form, automated checking by specialized detectors is ruled out.

Secondly, since the messages are in textual form and can be embedded in books, articles, newspaper classifieds, etc., the domain of filtering becomes too large to be checked in reasonable time. In other words, every text, be it present in an advertisement or a nursery rhyme, or even a retail invoice becomes a medium of concealment, and clearly, everything cannot be checked, not even with huge manpower, time and equipment. Thirdly, since the stego-text is generated manually, it would be both syntactically correct as well as meaningful. Not only each sentence, but the entire text would have the same context and would have a meaning of its own, unlike sentences generated by random generation methods. Therefore, the stego-text is unlikely to generate suspicion.

With respect to attacks, our proposed algorithm is not that robust. Changing of the words, or the order of the words, or even insertion of new words may alter the position vector and extraction would become impossible. The stego-text is only robust to attacks to the extent that irrelevant text may be appended at the end without affecting extraction. However, no capitalized words must be present in the appended text other than 'I' and the first word of every sentence. Insertion of text before the stego-text however will break the system.

4.4 Limitations and Solutions

There are some limitations of the proposed method, which we now state below, along with the proposed solutions to overcome them:

1. Expressing numbers (such as time, coordinates, etc.) are risky as they must also be present in the cover-text, but finding or generating suitable cover-text containing numbers, yet being innocuous, is cumbersome. The solution is to use retail invoices as the cover. The product rates/totals may be used to conceal the numbers, and the secret-text may be embedded in the 'Terms and Conditions' section that is usually present at the bottom of each invoice. The capitalized nouns may be very easily concealed in the product names mentioned in the invoice.

Since the words of the secret message must appear exactly in the cover-text, sensitive words may be too risky to send plainly. The solution is to break up the words and use partial-matching as shown before.

5. Conclusion

The steganographic approach discussed in this paper would find its application primarily in situations involving espionage on the ground with no means of digital communication at the disposal of the sender, and where even suspicion that a message is concealed in another message, would be fatal, irrespective of the fact that the secret message is extractable. The security of the proposed scheme lies in four factors: (i) its low-tech nature, (ii) meaningful and unsuspecting stego-text, (iii) no exchange of stego-keys between sender and receiver (unless encryption is specifically used), and (iv) exponential increase in the domain of filtering. The first and last factors are very important and are the ones which lend the most security to our scheme. Due to being low-tech, digital communication channels are not required and automated checkers cannot be used to detect concealment. Moreover, the stego-text can be generated by hand using basic tools (like pencil and paper), and can be manually copied as many times as required without losing the concealed information. Finally since the domain of filtering is exponentially large, almost every piece of text everywhere needs to be checked; which is impossible.

References

[1] M. Conway, "Code wars: Steganography, signals intelligence, and terrorism," Knowledge, Technology and Policy (Special issue entitled: Technology and Terrorism), vol. 16, no. 2, pp. 45-62, 2003.

[2] D. Denning and W. E. Baugh, "Hiding crimes in cyberspace," in Crypto Anarchy, Cyberstates, and

Pirate Utopias (P. Ludlow, ed.), Cambridge MA: MIT Press, 2001.

[3] "The encrypted jihad." http://www.salon.com/tech/feature/2002/02/04/terror encryption/, 2002. Online; retrieved on 30 March 2018.

[4] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," IBM Systems Journal, vol. 35, no. 3,4, pp. 313-335, 1996.

[5] M. Kwan, "The snow home page." http://www.darkside.com.au/snow/, 1998. Online; Retrieved on 20 March 2018.

[6] L. Y. Por, T. F. Ang, and B. Delina, "Whitesteg: a new scheme in information hiding using text steganography," WSEAS Transactions on Computers, vol. 7, no. 6, pp. 735-745, 2008.

[7] R. S. R. Prasad and K. Alla, "A new approach to telegu text steganography," in IEEE Symposium on Wireless Technology and Applications, IEEE, 2011.

[8] I. Banerjee, S. Bhattacharyya, and G. Sanyal, "Novel text steganography through special code generation," in Proceedings of the International Conference on Systemics, Cybernetics and Informatics, 2011.

[9] M. Shirali-Shahreza, "Text steganography by changing words spellings" in Proceedings of the 10th International Conference on Advanced Communication Technology, vol. 3, IEEE, 2008.

[10] M. D. Khairullah, "A novel text steganography system using font color of the invisible characters in Microsoft Word documents," in Proceedings of the 2nd International Conference on Computer and Electrical Engineering, vol. 1, IEEE, 2009.

[11] S. Roy and M. Manasmita, "A novel approach to format based text steganography," in Proceedings of the International Conference on Communication, Computing & Security, ACM, 2011.

[12] M. H. Shirali-Shahreza and M. Shirali-Shahreza, "A new approach to Persian/Arabic text steganography," in Proceedings of the 5th IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse, Computer and Information Science, IEEE, 2006.

[13] M. Chapman, G. I. Davida, and M. Rennhard, "A practical and effective approach to large-scale automated linguistic steganography," in Proceedings of the International Conference on Information Security, Springer, 2001.

[14] M. H. Shirali-Shahreza and M. Shirali-Shahreza, "A new synonym text steganography," in Proceedings of the IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IEEE, 2008.

[15] C.-H. Lin and T.-C. Lee, "A confused document encryption scheme and its implementation," Computers & Security, vol. 17, no. 6, pp. 543-551, 1998.

[16] W.-H. Yeh and J.-J. Hwang, "Hiding digital information using a novel system scheme," Computers & Security, vol. 20, no. 6, pp. 533-538, 2001.

[17] K. Bennett, "Linguistic steganography: Survey, analysis, and robustness concerns for hiding information in text," CERIAS Technical Report, 2004.

[18] N. F. Johnson, Z. Duric, and S. Jajodia, Information Hiding: Steganography and Watermarking - Attacks and Countermeasures, ch. 1, p. 4. Advances in Information Security, Springer, 2001.

[19] D. Kahn, The Codebreakers. Macmillan, 2nd ed., 1996.

[20] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (BlowFish)," in Proceedings of the Cambridge Security Workshop on Fast Software Encryption, pp. 191-204, Springer-Verlag, 1993.

[21] W. E. Winkler, "The state of record linkage and current research problems," Statistics of Income Division, Internal Revenue Service Publication R99/04, 1999.

[22] R. Anderson, R. Needham, and A. Shamir, "The steganographic File system," in Proceedings of the 2nd International Workshop in Information Hiding (D. Aucsmith, ed.), vol. 1525 of Lecture Notes in Computer Science, Springer-Verlag, 1998.

[23] T. G. Handel and M. T. Stanford, "Hiding data in the OSI network model," in Proceedings of the 1st International Workshop on Information Hiding (R. Anderson, ed.), vol. 1174 of Lecture Notes in

Computer Science, pp. 23-38, Springer-Verlag, 1996.
[24] C. H. Rowland, "Covert channels in the TCP/IP protocol suite," 1996.
[25] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Fingerprinting digital circuits on programmable hardware," in Proceedings of the 2nd International Workshop on Information Hiding (D. Aucsmith, ed.), vol. 1525 of Lecture Notes in Computer Science, Springer-Verlag, 1998.
[26] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Enhanced intellectual property protection for digital circuits on programmable hardware,"in Proceedings of the 3rd International Workshop on Information Hiding (A. Pfitzmann, ed.), vol. 1768 of Lecture Notes in Computer Science, Springer-Verlag, 2000.

Authors' Profile

Akash Nag is a faculty member at M.U.C. Women's College, Burdwan in the Department of Computer Science, where he teaches at the undergraduate level. He received his Masters degree in Computer Science from the University of Calcutta and his Ph.D. from The University of Burdwan. His main research interests include bioinformatics, algorithms, security, IoT, and wireless sensor networks.

How to cite this paper: Akash Nag,"Low-Tech Steganography for Covert Operations", International Journal of Mathematical Sciences and Computing (IJMSC), Vol.5, No.1, pp.18-30, 2019.DOI: 10.5815/ijmsc.2019.01.02