A Proposed Algorithm for Steganography

A.O. Ibadi, O. Z. Akif
Department of Software, College for Economic Sciences Engineering , University of Baghdad
Department of Computer Science, College of Education Ibn Al-Haitham, University of Baghdad

Abstract

Steganography is an important class of security which is widely used in computer and network security nowadays. In this research, a new proposed algorithm was introduced with a new concept of dealing with steganography as an algorithmic secret key technique similar to stream cipher cryptographic system. The proposed algorithm is a secret key system suggested to be used in communications for messages transmission steganography.

Introduction

The rise of the Internet and multimedia techniques in the mid-1990s has prompted increasing interest in hiding data in digital media. Early research concentrated on watermarking to protect copyrighted multimedia products (such as images, audio, video, and text). Data embedding has also been found to be useful in covert communication, or steganography. The goal was and still is to convey messages under cover, concealing the very existence of information exchange[1].

The problem in the methods used in the steganography is the ability to detect the information which embedded in the message by using the roles of detection. In this research, we use a new approach to solve this problem, the power of this approach is the ability to hide the information in random positions in the image that means any roles that's using in the detection can not succeed to detect any information that was embedded in the image thus a new approach prevent any ability for detection.

All previous methods do not use the random generator to hide the information, but for cryptography it is used. In a new approach we use the random generator to hide the information in the image by generate random positions from the image to hide the information on it.

The main attribute to this approach is to keep the same proprieties of the image before and after the embedding of the information such so that one cannot detect this image to have any information.

Steganography

We all know what is the purpose of steganography: to hide data into other data. The point of cryptography is to transform structured and intelligible data (like a text file) into a stream of random-looking bytes. The point of steganography is somehow the opposite, to mix random-looking data with decoy information so that it will look like structured format.

In the field of steganography, some terminology has developed. The term "cover" is used to describe the original, innocent message, data, audio, still video and so on. When referring to audio signal steganography, the cover signal is sometimes called the "host" signal.

The information to be hidden in the cover data is known as the "embedded" data. The "stego" data is the data containing both the cover signal and the "embedded" information. Logically, the processing of putting the hidden or embedded data, into the cover data, is sometimes known as embedding.

Occasionally, especially when referring to image steganography, the cover image is known as the container[2].

There are plenty of softwares around that can hide data on BMP images. Unfortunately, BMP pictures are not widely used or exchanged, unlike JPG. So why programmers don't perform stegano programs for JPG. The reason is more conceptual. The concept of lossy compression like the one used in JPG (or MP3 for audio) is to remove most of the unimportant or redundant information. The concept of most steganography algorithms is to hide bits by replacing this very same unimportant or redundant information (like the Least Significant Bits). So both techniques are going in opposite directions. The more you compress, the more difficult it is to find room to hide data.

Steganography classifications

We would classify steganography softwares in several categories of increasing quality. It's a little bit artificial as a scale, not absolutely defined with golden rules, but I would say it's a pragmatic way of quickly estimating software. Notice that I am not taking in account any upstream encryption, before the steganography step. That's another matter, although generally, programs that offer some solid, known and published encryption algorithm, should be trusted

more than any "in-house" obfuscation method. So the techniques are, one of the following classification:

1. Adding data at the end of the carrier file.
2. Inserting data in some junk or comment field in the header of the file structure.
3. Embedding data in the carrier byte stream, in a linear, sequential and fixed way.
4. Embedding data in the carrier byte stream, in a pseudo-random way depending on a password.
5. Embedding data in the carrier byte stream, in a pseudo-random way depending on a password, and changing other bits of the carrier file to compensate for the modifications induced by the hidden data, to avoid modifying statistical properties of the carrier file.

In our point of view, we don't consider points 1 and 2 as real steganography. Of course, what "real steganography" may depend on the definition you use. If we choose to use "hiding data from my young brother" as a definition, yes, this is steganography. The concept of breaking a steganography software does not mean that you can recover the hidden data. To break a steganography algorithm means that you can decide with a statistically high level of confidence that an image contains embedded information, and estimate the size of this information.

Least Significant Bit (LSB) insertion method

Least significant bit insertion is a common, simple approach to embed information in a cover file. The LSB is the lowest order bit in a binary value. This is an important concept in computer data storage and programming that applies to the order in which data are organized, stored or transmitted.

The last bit of the byte is selected as the least significant bit (as illustrated in Figure 1) because of the impact of the bit to the minimum degradation of images. The last bit is also known as right-most bit, due to the convention in positional notation of writing less significant digit further to the right.[3]

In this method, we can take the binary representation of the hidden data and overwrite the LSB of each byte within the cover image. If we are using 24-bit color, the amount of change will be minimal and indiscernible to the human eye.

As an example, suppose that we have three adjacent pixels (nine bytes) with the following RGB encoding:

10010101 00001101 11001001 10010110 00001111 11001010 10011111 00010000 11001011

Now suppose we want to "hide" the following 9 bits of data (the hidden data is usually compressed prior to being hidden): 101101101. If we overlay these 9 bits over the LSB of the 9 bytes above, we get the following (where bits in bold have been changed):

10010101 00001100 11001001 10010111 00001110 11001011 10011111 00010000 11001011

Note that we have successfully hidden 9 bits but at a cost of only changing 4, or roughly 50%, of the LSBs. This description is meant only as a high-level overview. Similar methods can be applied to 8-bit color but the changes are more dramatic. Gray-scale images, too, are very useful for steganographic purposes.[4]

Detection of Steganography

Steganalysis is the art and science behind the detection of the use of steganography by a third party. The basic function of steganalysis is to detect first or estimate the probability that hidden information is present in any given file. The detection and estimation are based only on the data presented in its observable form (i.e. nothing is known about the file prior to investigation). Because simply detecting the presence of hidden data may not be sufficient, steganalysis also covers the functions of extracting the message, disabling and/or destroying the hidden message so that it cannot be extracted, and finally, altering the hidden message such that misinformation can be sent to the intended recipient instead of the original message [5].

Depending on how much information is known about the embedded image, steganalysis techniques and methods closely mirror traditional cryptanalysis methods. The steganalysis attack methods can be broken into six types:

- Steganography-only attack: Only the file with the embedded data is available for analysis.
- Known-carrier attack: Both the original carrier file and the final (hidden message embedded) files are available for analysis.
- Known-message attack: The original message prior to embedding in the carrier is known.
- Chosen-steganography attack: Both the algorithm used to embed the data and the final (hidden message embedded) file are known and available for analysis.

- Chosen-message attack: The original message and the algorithm used to embed the message are available, but neither the carrier nor the final (hidden message embedded) file are. This attack is used by the analyst for comparison to future files.
- Known-steganography attack: All components of the system (the original message, the carrier message, and the algorithm) are available for analysis.

It follows that the success of any steganalysis technique is tied to the amount of information known about the file prior to investigation. As more information about the file is known prior to investigation, the investigator can move from simply detecting to modifying or altering the hidden message before sending it on to the intended recipient. In the first category (steganography-only attack), the purpose of analysis is to detect simply the existence of a hidden message.

Without prior knowledge of the encoding mechanism, key, or data contained within the message, recovery of the contents using this method while possible, can take an excessive amount of time. With access to the original carrier and the final file with the embedded content (known-carrier), the purpose of analysis can move toward recovering the embedded message by comparing the differences between the two files. If the algorithm is known and the file with the hidden message embedded is also available (chosen-steganography attack), the analyst may have the ability to reverse the embedding to recover the hidden message and can easily alter or destroy the hidden contents [5].

Finally, if the analyst has the algorithm and a message prior to embedding (chosen-message attack), they can move towards identifying possible (hidden message embedded) files to attempt to recover the original carrier. If the carrier can be recovered or closely reproduced, the ability to insert alternate messages in lieu of the original message is possible.

The steganography-only attack can be accomplished through the use of statistical analysis performed on the final medium. In the following example, the color contents of JPEG images are examined. A modification to each coefficient LSB produces variations in the data that results in deviations to the histogram for the given file. If the deviations are large enough to produce noticeable aberrations, the embedded file histogram can identify the existence of the hidden message. Likewise, LSB modifications to palette-based images (GIF, etc.) cause duplications of the colors in the palette with identical or nearly identical colors appearing. This duplication of colors can also serve as an indicator pointing to the existence of hidden data.

New proposed stego-system

The main concept for suggesting the proposed system is to keep stego-data out of detection, in other words, the detection test must fail. In the following sections a new proposed stego-system will be introduced starting with the general structure of the whole system and then the complete algorithm followed by sections to explain some which is not well-defined parts (new concepts) of the algorithm.

1.The General Structure

Figure 2, shows the general structure of the stego-algorithm. A message will be fed to the stego-algorithm to be embedded in well suited cover (image) of length enough to hold the embedded text.

For LSB techniques the length of the cover (L_{cov}) file must accomplished the following:

$L_{cov} \geq (N+55)*8$ bits

Where N is the length the embedded text in bits. This inequality is computed for a cover image of bitmap type (BMP). The number 55 is length of the image header in bitmap image type The stego-algorithm must have a stego-key which is a seed key fed to the stego-algorithm to hide the text in the cover image and produce the stego-data. These operations are performed in the sender side while the receiver will retrieve the stego-data and feed it with the same stego-key to the stego-algorithm to obtain the embedded text.

It is very important to know that the stego-data is an image and must be the same cover image which is used in the sender side, for this reason, the receiver doesn't need to obtain the cover image by extracting the embedded data from stego-data file.

2.Stego-Algorithm Concept

There are two approaches to keep the stego-data out of detection, the first is to hide a serial data in random positions of the cover image, and the second approach is to hide a random data in serial positions of the cover image. We can prove this consideration by applying the detection tests to the stego-data in later sections. The second approach can be applied by performing the transposition encryption algorithms to the embedded data before hiding them in the cover image, which means to apply two operations cryptography and steganography and this can be proved in other new research. In this paper, we proposed steganographic algorithm using the second approach.

To hide a serial bits in a random positions we require a pseudo-random numbers generator to work in a similar fashion as stream cipher encryption algorithms (see figure 3). The difference between the proposed system and stream cipher is that stream cipher uses the generated random bits to be Xored with the plain text bits while the proposed system uses these random numbers as a reference to a position to hide the embedded bit in the covered image.

The random numbers generator must be non-repeatable random numbers generator for a predefined range length or for the whole cover file length. As we mentioned in section 5, the hiding process will change roughly 50% of the LSBs, additionally, we mean to use part of the cover file to reduce the opportunity to detect the hiding information or break the stego-algorithm.

3.Pseudo Random Numbers Generator

The random number generator is a function fed with a seed value to generate a randomly selected numbers from a range defined by 2^n where n is length of the sequence bits that represents the seed value (will be discussed later). The generation function is non-linear and generated 2^n unrepeated random numbers.

The output number $(R)_{10} = (R_0R_1...R_{n-1})_2$ can be resulted from the input number $(I)_{10} = (I_0I_1...I_{n-1})_2$ at each step, where R, I are decimals and $R_0, R_1, ..., R_{n-1}, I_0, I_1, ..., I_{n-1}$ are binary digits as following:

$R_i=T \oplus \oplus I_i$ for i=0,1,…,n-1 , Where if i=0 then T=0, if i=1 then $T=I_0$ else $T=I_0*I_1*\ldots*I_{i-1}$.

Example: assuming I=178 what is the output of the function R.

The binary representation of I=10110010 so the output bits will be as following:

$R_0= 0 \oplus 0 = 0,$

$R_1= 1 \oplus 0 = 0,$

$R_2= 0 \oplus 1 \oplus 1 = 0,$

$R_3= 0 \oplus 1 \oplus 1 = 0,$

$R_4= 0 \oplus 1 \oplus 0 = 1,$

$R_5= 0 \oplus 1 \oplus 0 = 1,$

$R6=0 \oplus 0 \oplus 0 = 0,$

$R_7= 0 \oplus 0 \oplus 0 = 1,$

$R=(00011101)_2=(13)_{10}$

So, if the input I=178 the output will be R=13 to produce another output we must feed the last output (13) as input to the function and so on. The first five output will be: R_0=13, R_1=242 ,R_2=107 ,R_3=138 ,R_4=53 ,R_5=202 ,…

To show that this function produce unrepeatable sequence of random numbers in the range of 2^n where n is the number of bits to represent the block or the cover file length, assume that n=4 and the seed number is 11 the output will be

1011, 0000, 1111, 0111, 1000, 0011, 1100, 0101, 1010, 0001, 1110, 0110, 1001, 0010, 1101, 0100,1011, …

The seed 1011 will be repeated after (2^4=16) step and there are no repeated number in this range as been observed.

4.Seed Value Length

Seed value length (***N***) is the number of bits needed to convert the seed value into binary. This number (***N***) is very important and must be selected carefully. The importance of it obtained from being ***N*** represents the range of the random number generator which must be large enough to increase the complexity of the setgo- algorithm (***N*!**) and in the same time must not exceed the length of the cover file.

Basically, the first important process which must be performed is to calculate the value of (***N***) as following:

$N = \lceil \log_2 Len \rceil$ where ***Len*** is the length of the cover file and $\lceil \ \rceil$ means the nearest upper integer value.

N in our proposed algorithm limits the text length of the hiding information which must not exceed the value of ***N***. In other words, we must select the cover file as big enough to hide all the secret text. Practically, $N \geq 10$ is fair because 10! is large enough for not breaking the system. This means that the minimum cover file size must be greater than (2^{10}=1024) and other cases depending on the secret text length.

5.The Complete Structure

Figure 4, shows the complete structure of the proposed stego-system. Starting from reading the text file (which is wanted to be hidden) and the cover file and then performing some important preprocessing steps like calculating the value of ***N*** and the length of the hiding text (**Len**) and delimiting the secret text by putting the end delimiter at the end of the text (appending the word "END" or "STOP" at the end) to know where to stop retrieving the hiding text later,

then checking that **Len** is not greater than (2^N) which means that the cover file length is large enough to hide the secret information.

Randomness Tests

Randomness tests (or tests of randomness) are used to analyze the distribution pattern of a set of data. In stochastic modeling, as in some computer simulations, the expected random input data can be verified to show that tests were performed using randomized data. In some cases, data reveals an obvious non-random pattern, as with so-called "runs in the data" (such as expecting random 0-9 but finding "4 3 2 1 0 4 3 2 1..." and rarely going above 4). If a selected set of data fails the tests, then parameters can be changed or other randomized data can be used which does pass the tests for randomness[6].

There are many practical measures of randomness for a binary sequence. These include measures based on statistical tests, transforms, and complexity or a mixture of these. The best for testing the semi-random generators are based on statistics which are as follows [7]: the frequency test, the serial test, the poker test, the run test and the auto correlation test , which all depend on X^2 distribution

Conclusions

To conclude that the key generator is a random generator we need to prove that it can pass the randomness tests.

Figure 5, shows the test results of a several generated output sequences of a different length, All of them passed the randomness tests.

References

1. HUAIQING WANG AND SHUOZHONG WANG, (2004),Cyber Warfare: Steganography vs. Steganalysis", October / 47(10), COMMUNICATIONS OF THE ACM.
2. www.wikipidia.com, The free encyclopedia, steganography.
3. Por, L.Y.;. Alireza,W. K. Lai2 Z. ; Ang,F. T.and Su,M.T. B. "StegCure: A Comprehensive Steganographic Tool using Enhanced LSB Scheme" Delina, Faculty of Computer Science and Information Technology University of Malaya 50603, Kuala Lumpur MALAYSIA.

4. Murshed ,Md. M. "Steganography using LSB hiding" Upper Iowa University Fayette, Iowa, USA.

5. Dickman,S. D. and Madison, J. (2007).An Overview of Steganography" University Infosec Techreport Department of Computer Science JMU-INFOSEC-July

6. www.wikipedia.com/randomness tests.

7. Robshaw,M.J.B. (1995),Stream Ciphers, RSA Laboratories Technical Report TR-701, Version 2.0|July 25,.

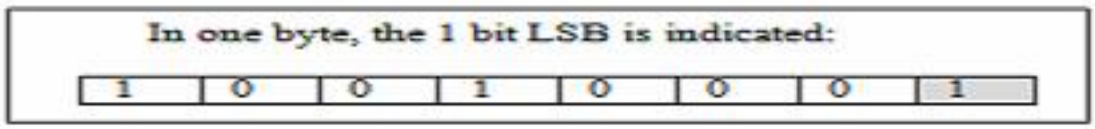

Fig.(1): Least Significant Bit

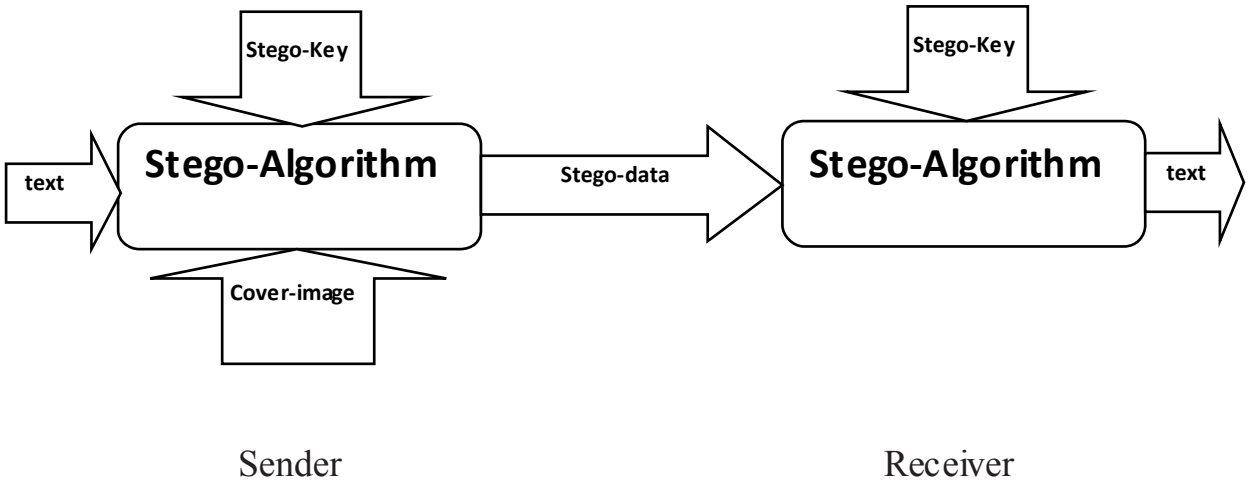

Fig.(2): the general structure

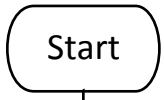

Fig.(4): The Complete Structure of the Stego-system

Sequence length	**Frequency Test** (≤3.84)	**Serial test** (≤ 7.81)	**Poker Test** (≤ 11.1)	**Run Test** T0 gaps test T1 blocks test	**Autocorrelation test (d=8)** (P=pass, F=fail)
1024	0.34	0.88	8.01	T0=pass T1=pass	PPPPPPPP
2048	2.11	1.4	6.41	T0=pass T1=pass	PPPPPPPF
4096	3.01	1.44	8.44	T0=pass T1=pass	PPPPPPPF
8192	3.00	3.45	3.6	T0=pass T1=pass	PPPPPPPF
16384	0.78	2.23	2.19	T0=pass T1=pass	PPPPPPPF

Fig.(5): Randomness tests results

مجلة ابن الهيثم للعلوم الصرفة والتطبيقية المجلد23 (3) 2010

خوارزمية مقترحة لأخفاء البيانات

عبدالكريم عكلة،عمر زياد عاكف
قسم هندسة البرامجيات ، كلية بغداد للعلوم الاقتصادية
قسم علوم الحاسبات، كلية التربية– أبن الهيثم ، جامعة بغداد

الخلاصة

يعد علم أخفاء البيانات أحد الفروع المهمة لعلم التشفير الذي يستخدم بصورة واسعة في أمنية الحاسبات والشبكات في هذه الايام. في هذا البحث اقتراحت خوارزمية جديدة تم تقديمها بمفهوم جديد للتعامل مع علم اخفاء البيانات وفي تقنية حساب المفتاح السري مشابهة لنظام التشفير الانسيابي. الخوارزمية المقترحة هي مفتاح سري تم أقترح لكي يستخدم في الاتصالات لغرض أخفاء معلومات الرسائل المرسلة.