Detection of Hidden Encrypted URL in Image Steganography

Moudhi Aljamea, Tanver Athar, Costas S. Iliopoulos, M Samiruzzaman
Department of Informatics,
Kings College London,
WC2R 2LS, London
Email: [mudhi.aljamea@kcl.ac.uk, tanver.athar@kcl.ac.uk, c.iliopoulos@kcl.ac.uk,
mohammad.samiruzzaman@kcl.ac.uk]

***Abstract*—Steganography is the science of hiding data within the data, either for the good purpose of secret communication or for the bad intention of leaking confidential data, embedding malicious code or Uniform Resource Locator (URL). Various carrier file formats can be used to hide this data (network, audio, image etc.). The most common steganography carrier is embedding secret data within images. We can hide different formats (another image, text, video, virus, URL etc.) inside an image. To the human eye, the changes in the image appearance with the hidden data can be imperceptible. This paper proposes an implementation of a novel detection approach that will concentrate on detecting any kind of hidden URL in most types of images and extract the hidden URL from the carrier image using the Least Significant Bit (LSB) hiding technique. We have recently introduced an algorithm for *Detection of URL in Image steganography*. In addition, we have extended the algorithm to detect and extract encrypted URLs. In this paper, implement the proposed algorithm, successfully test it and compare it with various results, using different images.**

Keywords–Steganography; Image Steganography; Security; String Matching; Steganalysis; URL Detection.

I. INTRODUCTION

Steganography is the science of hiding data within data. The word steganography is derived from the Greek words stegos, meaning cover, and grafia, meaning writing [1]. There are some differences between steganography and cryptography. Cryptography is the art of scrambling messages to make them difficult to understand, whereas steganography is the art of hiding messages to make them difficult to find. Therefore, steganography is an extra layer that will support transferring secret information securely whereas, cryptography, in this case, is data protection. Besides, when steganography fails and the message can be detected, it is still of no use as it is encrypted using cryptography techniques [2].

Steganographic techniques started in ancient Greece. One early example consisted in writing text on wax-covered tablets. Another example involved shaving the head of a messenger and tattooing a message or an image on the messenger's head and let the hair grow back. The message would remain undetected until the head was shaved again [3].

The science of steganography has developed significantly to more sophisticated techniques, allowing a user to hide large amounts of information within images, audio files, and even networks. In fact, the main difference between the modern steganographic techniques and the previous ones is only the form of carrier for the secret information. Researchers are devising new steganographic applications and techniques and old methods are given new twists [3].

Our Contribution: We have recently introduced an algorithm for *Detection of URL in Image Steganography* [4]. In this paper, we extend the algorithm to detect and extract encrypted URLs. We implement the proposed algorithm and successfully test it and compare it to various results using different images.

Structure of the paper: In Section II, we present some background related to image steganography. In Section III, we discuss stegonalysis, which is the main component of detecting hidden messages inside the image. In Section IV, we discuss the problem of hiding URLs inside an image. In Section V, we discuss and present the algorithm, algorithm complexity analysis, implementation and results of the experiments. In Section VI, we present the conclusion and future work.

A. The Concept of Steganography

The concept of steganography is to embed data, which is to be hidden. However, this process will require three files:

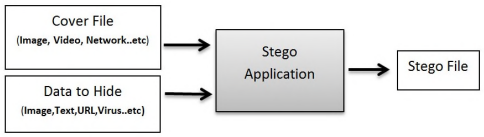

Figure 1. Stego application scenario

First, we have the secret message, which is the information to be hidden and, as mentioned before with the new steganography techniques, almost any kind of data can be hidden. *Second,* we have the cover file that will hold the hidden information and, similarly, almost any kind of file can be used as a carrier. *Finally,* we have the key file to find the hidden message and extract it from the cover file. The result of these three files is a file called stego file, as shown in Fig. 1.

The most common steganography technique is embedding messages within images, as it is considered the best carrier to hide all types of files within it. For example, it is possible to hide another image, virus, URL, text, exe file, audio etc. without changing its visible properties [5].

 ISBN: 978-1-61208-534-0

B. Steganography Applications

Steganography can be used in many useful ways. For example, to help in transferring secret data, copyrights control of materials and smart identity cards (IDs), where individuals' details are embedded in their photographs [6]. It can be used in printed images, where the data can be embedded after printing the image. The user can scan the printed image with a smart device and the embedded information will appear on his/her device. This technique is useful in exhibitions and displaying the product's information.

Cybercrime is believed to benefit from steganography in transferring illegal data or embedding viruses and malicious URLs. To counter this threat, new techniques and methods are being developed and this area is getting more attention among researchers.

There are many sophisticated steganography pieces of software available online, which can be used for cybercrime. Xiao steganography [7] is one such tool. Any user can use this tool to leak his/her company's confidential information.

For this reason, many companies are finding it difficult to detect the stego files even after scanning all their employees outgoing emails.

II. IMAGE STEGANOGRAPHY

Images can be more than what we see with our eyes. To use an image as a cover file is considered to be one of the most useful and cost effective techniques [8]. All image steganographic techniques to hide data are based on the structure of the most commonly used image format on the Internet: graphics interchange format (GIF), portable network groups (PNG) and Bit Map Picture (BMP).

- Cover Image: In steganography, the original image that was chosen as a carrier for the secret data is called a cover image.
- Stego Image: This is the result image of choosing the right cover image and embedding the secret data inside it.
- Stego Key: The sender should have an algorithm for create the stego image to embed the data, and the receiver should have the matching algorithm to extract the hidden data from that particular stego image. Sometimes, the process will require a key, which is similar to a key used in cryptography, to extract the hidden message, and that key is called stego key.

Image Embedding Process: Let C be the chosen cover image, and C' be the stego image, K be the stego key, and M be the hidden message then:

$$C \oplus M \oplus K \rightarrow C'$$

as shown in Fig. 2.

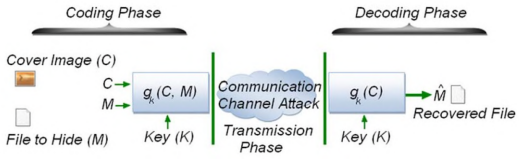

Figure 2. Image Steganography Embedding process

The main challenge in image steganography is that many image manipulation techniques might destroy the hidden message on any image, since it will change the feature of the stego image. Cropping might destroy or corrupt part of the hidden message if the hidden image is located where the image is cropped. Rotation might give the receiver difficulty in finding the hidden message. Filtering might destroy the hidden message completely.

A. Current Image Steganography Techniques

There are some naive implementations of image steganography, such as by feeding windows operating system (OS) command some code to embed the text file which contains the secret message into a specific image and produce the stego image (Fig. 3).

```
C:> Copy Cover.jpg /b + Message.txt /b Stego.jpg
```

Figure 3. Stegocode

Steganography embedding techniques can be divided into two groups. The first is the Spatial Domain, also known as Image Domain, which embeds the secret data directly in the intensity of the image pixels, usually the Least Significant Bit (LSB) in the image. The other is the Transform Domain, which is also known as Frequency Domain, where images are first transformed and then the secret data is embedded in the image [9].

The focus of this paper will be on the spatial domain. In the spatial domain, the steganographer modifies the secret data and the cover image, which involves re-encoding the LSBs in the carrier image. To the human eye, these changes in the image value of the LSB are imperceptible [10]. This technique can be applied for most image formats.

Least Significant Bits: This technique embeds bits of the secret data directly into the LSB plane of the cover image [6]. LSB is considered to be one of the simplest approaches of embedding data in a cover image. Yet, it is one of the most difficult approaches to detect.

On average, the changes will be only made on three bits with a byte. Only half of the bits within an image are modified to hide the secret data using the maximal cover size. The result of these changes is too small to be recognized by the human visual system (HVS) [1].

 ISBN: 978-1-61208-534-0

III. STEGAANALYSIS

Steganalysis is the main step in the steganography technique to discover the hidden messages. It is the way of identifying the suspected medium, determine whether or not they have an embedded data into it, and, if possible, recover that data. Steganalysis is the science of attacking steganography in a battle that never ends [6].

Steganlysis can sometimes be more challenging than cryptanalysis. The steganalyst first has to identify the suspected cover file, then locate the hidden message. The hidden message might be scattered in different locations inside the cover file. In some cases, the hidden message might be encrypted to make it more difficult to detect. The main mission of the cryptanalyst is to decrypt the encrypted message.

There are 4 types of Steganalysis listed below:

1) **If the steganography attack is known to the stegnalysis:** since the cover file, the hidden message and the steganography tool (algorithm) are all known to the steganlysis, the hidden message can be identified quite easily.
2) **Only the original file (before embedding the message) and the cover file are known to stegnalysis:** the objective will be to compare the two files, and using pattern difference between the two files, to identify the hidden message.
3) **If only the secret message is known to the stegnalysis:** the objective is to identify a known pattern in all the files. This is a difficult approach.
4) **Only the cover file is known to the stegnalysis:** similarly to the previous attack, it can be very challenging to identify the hidden message location, since it may be scattered to more than one place.

Image analysis forms the backbone of the image steganalysis programs. Image manipulations techniques, such as translating, filtering, cropping and rotation are used in steganalysis. Discrete cosine transform (DCT)-based image steganography hints can be identified using JPEG double compression and the DCT transform [6].

The focus of this paper will be a new kind of attack where the type of the hidden message is URL and the hiding technique LSB are both known.

A. URL in Image Steganography

Embedding data in images is not a new technique. This method is getting better and more sophisticated. One of the recent improvements is embedding a URL in the image LSBs (see Fig. 4). The objective of the URL is to direct the receiver to a web page. The web page might contain a virus that will harm the image receiver, either by destroying or stealing data.

The main reason behind embedding an URL in an image instead of the whole secret data is that the URL requires very little space in the carrier [11]. This ensures that the URL can be difficult to detect and there is less chance of losing the URL by image manipulations.

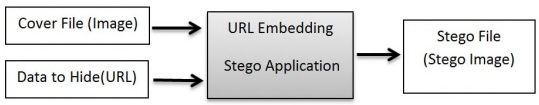

Figure 4. URL Stego Embedding Scenario

In [12], the authors discusses about stegoloader malware. It was noted that the malware authors are evolving their techniques to evade network and host-based detection mechanisms. Stegoloader could represent an emerging trend in malware, hiding malicious code inside a digital image. Stegoloader has a modular design and it uses digital steganography to hide its main module's code inside a legitimate PNG image.

One malware, Lurk Downloader [12] specifically embeds URLs into an image file by inconspicuously manipulating individual pixels. The resulting image contains additional data that are virtually invisible to an observer. Lurk's primary purpose is to download and execute secondary malware payloads [13].

IV. THE PROBLEM

There are various types of information that can be hidden in the LSB of an image. In this paper, we are dealing with an URL hidden inside an image. Any malicious code can be embedded by using LSB. To modify the LSB means to modify the colour, by using LSBs of an image.

There are different colour ranges which require different amounts of memory, such as 2 bits, 8 bits, 24 bits etc. They have both colour and grey scale. 8 bits colour means each pixel can have any of 256 (2^8) colours. The same calculation is applicable 8 bits grey scale or 24 bit colours. Since there are many colour combinations, modifying the LSB does not make much difference to the human eye. URL attack uses this weakness in colour LSB.

For example, an URL "http://exampleattack.com" has 24 characters. Each character of this URL takes 8 bits in ASCII format. The URL will require 192 significant bits from an image.

For simplicity, let us see how the first character 'h' of our example URL "http://exampleattack.com" can be added by using LSBs of an image. The ASCII value for 'h' is decimal 104 and binary 01101000

Before LSB insertion let us assume that 8 consecutive bytes of an image are:
10000010 10100110 11110101 10110101 10110011 10010111 10000100 10110001

After inserting 'h' (01101000) in LSBs, the result is below.
10000010 10100111 11110101 10110100 10110011 10010110 10000100 10110000

In this way, by using more significant bits of images, we can embed the rest of the characters of the intended URL.

 ISBN: 978-1-61208-534-0

V. URL DETECTION ALGORITHM

We are going to present an algorithm overview to detect a hidden URL from the LSBs of an image.

TABLE I. LIST OF TOP-LEVEL DOMAINS (TLD) BY THE ICANN FOR FULL LIST PLEASE REFER TO [14]

AAA	AARP	ABB	ABBOTT	ABOGADO
AC	ACADEMY	ACCENTURE	ACCOUNTANT	ACCOUNTANTS
ACO	ACTIVE	ACTOR	AD	ADS
ADULT	AE	AEG	AERO	AF
AFL	AG	AGENCY	AI	AIG
AIRFORCE	AIRTEL	AL	ALLFINANZ	ALSACE
AM	AMICA	AMSTERDAM	ANALYTICS	ANDROID
AO	APARTMENTS	APP	APPLE	AQ
AQUARELLE	AR	ARAMCO	ARCHI	ARMY
ARPA	ARTE	AS	ASIA	ASSOCIATES
AT	ATTORNEY	AU	AUCTION	AUDI
AUDIO	AUTHOR	AUTO	AUTOS	AW
AX	AXA	AZ	AZURE	..etc

A. Algorithm Overview

Step 1: Create a sorted list, DOMAIN[], from the static official top level domain list.

Step 2: Create an array called BITMAP[], from an image taking each bit in the array.

Step 3: Make a character array called, LSBCHARARRAY[] from an intermediate array of LSBARRAY[] by converting each 8 bits to an ASCII character.

Step 4: Loop through the LSBCHARARRAY[], find out possible hidden URL is formed by http or https, www, domain name and TLD.

B. Complexity Analyses

1) Step 1 (Create a sorted list from the static official top level domain):: **Space complexity**: We have a known TLD list [14]. So, in the pre-processing stage, we have created an indexed array, DOMAIN[] considering each TLD as a string. It is linear to the size of all characters plus the index of each string position in a sorted order. We have created a separate index list with just starting position of TLDs with a specific character.

For example, if .co and .com both starts with c, so if we know where the c starts on the whole sorted list, we just can look at the block starts with 'c'. The overall space complexity of the sorted list is O(M) +O(t) + O(i), where M is the total number of characters, it is the index on each TLD string which is limited to the official static list.

Time complexity: This step of computation can be a pre-processing step, so complexity is not a major issue. However, it is possible to build up a sorted list by radix sort [15] where an LSD radix sort operates in $O(nk)$ in all cases, where n is the number of keys, and k is the average key length.

2) Step 2 (Create a sorted list from the static official top level domain list):: **Space complexity:** O(M) where M is the number of bits.

Time complexity: O(n) where n is the number of bits. This means in just a single iteration the array is built.

3) Step 3 (Make a character array by converting each 8 bits to an ASCII character):: **Space complexity**: The complexity is $O(n)$ here where n=M/8 where M is the number of bits in BitMap and only one in each 8 bits are placed in a character array by converting 8 such Least Significant bits into character. So, the complexity here is sub linear. Although an intermediate LSBARRAY has been introduced in Step 3 for clarity purpose of the flow, it is possible to calculate the LSBCHARARRAY directly from BITMAP[] array. So LSBARRAY[] is not required in the implementation.

Time complexity. This is looping through the BitMap array just once and producing a character array by taking each 8 significant bits together and converting to ASCII. So, the time complexity is linear here with O(n) where looping n bits just once produces the result. Converting to ASCII and character has happened just 1 in 1/64 where 1 byte (8 consecutive LSB) comes from 64 bits. This operation produces a time complexity of O(n+n/64) which is linear.

4) Step 4 (Loop through the array, find out possible hidden URL is formed by http or https, www, domain name and TLD): **Space complexity**: The space complexity is linear with O(n), where n is the number of characters in the array.

Time complexity: This is a loop through the character array. Finding the first 3 parts of an URL (http/https and/or www, domain name) are done in one go in the single loop. They are part of the inside loop, used to find the position and calculation purpose for the string 'http', 'https' and 'www'. The actual counter of the characters array is incremented in each go whether it is inner loop or outer loop. The complexity holds linear for the operations because the whole character array are traversed just once.

Looking at the 4th part, TLD requires a short lookup in a sorted array described in Step 1. For the whole character array, this lookup is just done to complete the search in a sorted and indexed Top Level Domain array which we called in step 1 as DOMAIN[]. In a sorted list, the binary search works as log(n) complexity in the worst case where n is the number of items in an array. But, in our case, n is narrowed down by the index of each character. So, each block of searched area is n/m, where m is the number characters in the alphabet. So, the search takes log(n/m)time because we know the starting character what to lookup DOMAIN[] array. The overall complexity stays linear for step 4.

C. Next Level Detection (Detecting and Extracting Encrypted URL)

The previous tool can be considered as one of the first tools to detect the hidden text in images and extract these hidden messages. We have taken the algorithm to the next level, to detect and extract encrypted URLs.

In this new algorithm, the sender will encrypt and store the URL using the NOT encryption technique in the LSB of the image.

 ISBN: 978-1-61208-534-0

The following proposed algorithm is a linear time algorithm so, it terms of time and space, it does not add any more complexity compared to the previous algorithm.

D. The NOT Encryption Technique

This level of text encryption will not be detectable using the previous algorithm, since it will evade the URL detection through using the binary operation NOT to encrypt the plain text.

We continue with the example that was mentioned in the Problem Definition section:

The ASCII value for '*h*' is decimal 104 and binary 01101000

Before the LSB insertion, let us assume that 8 consecutive bytes of an image are below.

10000010 10100110 11110101 10110101
10110011 10010111 10000100 10110001

To add the extra encryption level to the plain text before embedding it in the image, the '*h*' binary NOT will transform from 01101000 to 10010111

Therefore, after inserting the encrypted '*h*' (10010111) in the LSBs the result is below.

10000011 10100110 11110100 10110101
10110010 10010111 10000101 10110001

The strength of this technique is that it will encrypt the URL, which is a very short text embedded in a very large number of pixels. It gives the sender the advantage of hiding the text without any key for the receiver to use to extract the text. The receiver will only need to know the hiding technique and the text location to extract the hidden text. There are many encryption mechanisms. The complement is one of the easiest mechanism to encrypt data.

E. Experiments

The solution was implemented using Visual Studio 2015 Studio, ASP.Net 4.5 and javascript. The solution is available her [18]. It was tested using BMP, PNG and GIF images of different sizes, colour depth, colour palettes and compression types. The solution has been tested using different browsers such as IE11, Firefox 4 and Chrome Ver 50.0. We have used 2 dozen different images with image size ranging from 300 bytes to 10 KB, colour depth ranging 2 bits to 24 bits, colour palettes ranging from 2 to 65K.

It uses javascript as a client side scripting language and it will work only on the browser where javascript is enabled. It also needs to access files from client machines or folders, so if there are restrictions on accessing image files, the browser will not be able to read the image files.

It cannot accept compressed and lossy images as there is a possibility that the URL data will be lost or corrupted when the images are compressed and the solution will not be able to extract the URL from the stego image [9]. Furthermore, for monochrome images, changing the LSB technique might alter the image in such a way that the changes are visible to the viewers and raise suspicion that the image have been altered, therefore, it will be eliminated.

F. Checking Experiment Results

1) Image Difference: We tested the generated images with the original images using a free image comparison website [16]. The website found no difference between the original and the image containing the hidden URL. In comparing the pixel value and colour between the images, there is a threshold (3 points) which the pixel must exceed in order to register as a difference. It confirms that the statistics steganalysis techniques will not be effective in detecting and extracting the hidden encrypted URLs since they are very short and the changes that they do to the images are imperceptible.

2) Histograms Analysis: We have analysed the histograms of the original image and the generated stego image using the website [17]. There was no difference between the histograms of both the original image and the stego image. It also confirmed that the steganalysis depending on the histograms of images will not detect the hidden URL even if the original image is known and the stego image is known as well.

VI. DISCUSSION AND FUTURE WORK

This paper describes in detail the existing research on how data can be hidden in an image. It also explains how to extract the hidden URL detection in images and the new algorithm to detect encrypted URL.

The URL detection problem in images was simplified with respect to string matching approach, which can be used in other kind of string matching problems in an image. For example, users may be interested to search for malicious commands or other kind of strings hidden in the image using the LSB of the image. The proposed solution has taken the previous URL detection algorithm to the next level, detecting and extracting encrypted hidden URLs. We have implemented and successfully tested this new algorithm using different images.

The experiments showed that the solution is very effective in detecting and extracting the URLs.

Detecting and extracting URL as it is, specifically in images, is a novel approach in image steganography analysis. The reason of concentrating on this problem is a response to the introduction of the new technique of embedding malicious URLs in images recently, and that is relatively a new technique for hiding/spreading viruses. The approach time and space complexity are promising. Therefore, as a future work the detection tool can be improved to cover more encrypting techniques.

 ISBN: 978-1-61208-534-0

VII. REFERENCES

[1] M. Hariri, R. Karimi, and M. Nosrati, ”An introduction to steganography methods,” World Applied Programming, vol. 1, no. 3, 2011, pp. 191- 195

[2] R. Krenn, ”Steganography and steganalysis,” Retrieved September, vol. 8, 2004, p. 2007.

[3] N. F. Johnson and S. Jajodia, ”Exploring steganography: Seeing the unseen,” Computer, vol. 31, no. 2, 1998, pp. 26-34.

[4] M. M. Aljamea, C. S. Iliopoulos, and M. Samiruzzaman, ”Detection of url in image steganography,” in Proceedings of the International Conference on Internet of Things and Cloud Computing, ser. ICC 16. New York, NY, USA: ACM, 2016, pp. 23:1-23:6. [Online]. Available: http://doi.acm.org/10.1145/2896387.2896408

[5] N. Provos and P. Honeyman, ”Hide and seek: An introduction to steganography,” Security and Privacy, IEEE, vol. 1, no. 3, 2003, pp. 32- 44.

[6] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, ”Digital image steganography: Survey and analysis of current methods,” Signal pro-cessing, vol. 90, no. 3, 2010, pp. 727752.

[7] softonic. Xiao steganography. Accessed: 2017-01-15. [Online]. Available: http://xiao-steganography.en.softonic.com/ (2015)

[8] C. Mohapatra and M. Pandey, ”A review on current methods and application of digital image steganography.” International Journal of Multidisciplinary Approach and Studies, vol. 2, no. 2, 2015.

[9] T. Morkel, J. H. P. Eloff, and M. S. Olivier, ”An overview of image steganography,” in Proceedings of the Fifth Annual Information Security South Africa Conference (ISSA2005), H. S. Venter, J. H. P. Eloff, L. Labuschagne, and M. M. Eloff, Eds., Sandton, South Africa, 6 2005, published electronically.

[10] Y. J. Chanu, T. Tuithung, and K. Manglem Singh, ”A short survey on image steganography and steganalysis techniques,” in Emerging Trends and Applications in Computer Science (NCETACS), 2012 3rd National Conference on. IEEE, 2012, pp. 52-55.

[11] O. K. E. Satir, ”A distortionless image steganography method via url,” in The 7th International Conference Information Security and Cryptology, 2014.

[12] D.S.C.T.U.T. Intelligence Stegoloader: A stealthy information stealer.Accessed:2017-01-15. [Online]. Available: http://www.secureworks.com/cyber-threat-intelligence/threats/stegoloader-a-stealthy-information-stealer/ (2015)

[13] D.S.C.T.U.Brett Stone-Gross,Ph.D. Malware analysis of the lurk downloader. Accessed: 2017-01-15. [Online]. Available:http://www.secureworks.com/cyberthreat-intelligence/threats/malware-analysis-of-the-lurkdownloader/?view=Standard (2014)

[14] ICANN. List of top-level domains. https://www.icann.org/resources/pages/tlds-2012-02-25-en. [Online].Available:https://www.icann.org/resources/pages/tlds-2012-02-25-en (2016)

[15] R. Sedgewick and K. Wayne. Radix sorts. [Online]. Available: https://www.cs.princeton.edu/rs/AlgsDS07/18RadixSort.pdf (2014)

[16] J. Cryer. Image analysis and comparison. Accessed: 2017-01-15.[Online].Available: https://huddle.github.io/Resemble.js/ (2015)

[17] LunaPlc.com. Histogram of image colors.Accessed:2017-01-15. [Online]. Available: http://www169.lunapic.com/editor/?action=histogram (2017)

[18] http://tanvera-001-site2.htempurl.com (2017)

 ISBN: 978-1-61208-534-0