Detection Of URL In Image Steganography

Moudhi M Aljamea
Department of Informatics
Kings College London
WC2R 2LS, London
mudhi.aljamea@kcl.ac.uk

Costas S. Iliopoulos
Department of Informatics
Kings College London
WC2R 2LS, London
c.iliopoulos@kcl.ac.uk

M Samiruzzaman
Department of Informatics
Kings College London
WC2R 2LS, London
mohammad.samiruzzaman@kcl.ac.uk

ABSTRACT

Steganography is the science of hiding data within data. Either for the good purpose of secret communication or for the bad intention of leaking sensitive confidential data or embedding malicious code or URL. However, many different carrier file formats can be used to hide these data (network, audio, image..etc) but the most common steganography carrier is embedding secret data within images as it is considered to be the best and easiest way to hide all types of files (secret files) within an image using different formats (another image, text, video, virus ,URL..etc). To the human eye, the changes in the image appearance with the hidden data can be imperceptible. In fact, images can be more than what we see with our eyes. Therefore, many solutions where proposed to help in detecting these hidden data but each solution have their own strong and weak points either by the limitation of resolving one type of image along with specific hiding technique and or most likely without extracting the hidden data. This paper intends to propose a novel detection approach that will concentrate on detecting any kind of hidden URL in all types of images and extract the hidden URL from the carrier image that used the LSB least significant bit hiding technique.

Keywords

Steganography ; Image Steganography; Security ; String Matching; steganalysis ; URL Detection

1. 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]. Moreover, Steganography and cryptography considered to be complementing each other rather than replacing each other. Cryptography is the art of scrambling messages to make it difficult to understand. Whereas, Steganography is the art of hiding it to make it difficult to find. Therefore, steganography is an extra layer that will support transferring secret information in a secure way (Secrete Communication) 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].
Moreover, Steganographic techniques started ages ago back to ancient Greece. Starting by writing text on wax-covered tablets to shaving the head of a messenger and tattoo a message or image on the messenger's head, then, the hair will grew back, and the message will be undetected until the head was shaved again [3].
Since then, the science of steganography has developed significantly to more sophisticated techniques far more than their ancient predecessors, allowing a user to hide large amounts of information within image, audio files and even networks. In fact, in reality the main difference between the modern steganographic techniques and the previous once is only the form of (carrier) for the secret information. For instance, instead of using human skin and wooden tables they use media files like images and audio.
Although, within the daily discovery of a message hidden with an existing application, a new steganographic applications are being devised. And an old methods are given new twists[3]. Therefore, there are so many types of steganography methods to hide secret data wither with new carriers, new hiding techniques or new type of secret data.

1.1 The Concept of Steganography

The concept of steganography is to embed data, which is to be hidden, however, this process will require three files: First, is 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, is the cover file (carrier) that will hold the hidden information and as well almost any kind of files can be used as a carrier. Finally, 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 Figure 1.

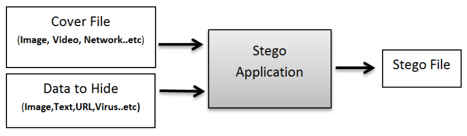

Figure 1: Stego application Scenario

WOODSTOCK '97 El Paso, Texas USA
© 2016 ACM. ISBN 978-1-4503-4063-2/16/03…$15.00
DOI: http://dx.doi.org/10.1145/2896387.2896408

However, the most common steganography technique is embedding messages within images as its considered to be the best carrier (Cover File) to hide all types of files within it. For example, hiding (another image, virus, URL, text, exe file , audio..etc) without changing its visible properties [4].

1.2 Steganography Applications

Steganography can be used for good intentions in many useful ways for example to help in transferring secret data, copy rights control of materials and smart IDs (identity cards) where individuals' details are embedded in their photographs [5], also it can be used in printed images where the data will be embedded before printing and after printing the user can scan the printed image with a smart device and the embedded information will appear on the device, that can be useful in so many fields especially exhibitions and as a marketing tool to display the products information.
Nevertheless, hospitals are using Steganography also to keep their patient's confidential data such as DNA sequences in a secret safe place where the access to it is highly restricted unlike other data.
On the other hand, like any other science, cyber-crime is believed to benefit from it in transferring illegal data or embedding viruses and malicious URLs in carries and other harmful actions. Therefore, due to the rapid development of steganography methods and techniques the steganography research area has gained so much attention during the last decade.
Besides that, nowadays any person without any technical background can do harmful cyper-crime actions with the support of the cyper-crime ready made sophisticated softwares which are available online to everyone with no cost and easy to use, for example there are alot of steganography tools for instance, Xiao Steganography [6] which any user can use it to leak his/her company's confidential information with basically 3 clicks (selecting the cover image, selecting the secret file or typing the secret message, and finally clicking on the embedding button) and the software will do the rest.
For this reason, many companies are finding it difficult to detect the stego files even after scanning all their employees out going emails.

2. IMAGE STEGANOGRAPHY

Images can be more than what we see with our eyes. Using an image as a cover file is considered to be one of the most useful and cost effective technique [7], all the image steganographic techniques to hide data based on the structure of the most commonly used images format on the internet (GIF-graphics interchange format), (JPEG-joint photographic expert group), (PNG-portable network groups) and (BMP- Bit Map Picture).

- ***Cover Image***: In steganography the original image that was chosen as a carrier for the secret data is called a cover image.
- ***Stego Image***: 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 creating 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 and its called stegokey.

Image Embedding Process : Let C be the chosen *Cover Image*, and C' is the *Stego Image*, the *Stego Key* will be denoted as K, and the hidden message as M then:
$C \oplus M \oplus K \rightarrow C'$
as shown in Figure 2.

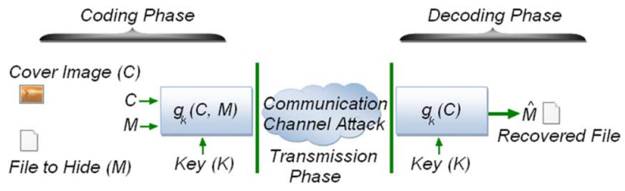

Figure 2: Image Steganography Embedding process [5]

However, 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) it might as well change the feature of the hidden message inside it, such as cropping might crop the hidden message if it was located in one section of the image or corrupt it, rotation might give the receiver difficulty in finding the hidden message, filtering might destroy the hidden message completely and so on.

2.1 Current Image Steganography Techniques

There are some naive implementation of image steganography for example by feeding windows OS command some code to embed the text file which contain the secret message into a specific image and produce the stego-Image.

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

Figure 3: stegocode

However, when displaying the image structure the message reveals itself and will not survive any kind of image manipulation.

Steganography embedding techniques can be divided into two groups either Spatial Domain also known as Image Domain which embed directly the secret data in the intensity of the image pixels usually the least significant bit (LSB) in the image, or 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 [8].
According to (A Review on current Methods and application of Digital image Steganography 2015) which present and study the major steganography algorithms between the year of 2010 and 2014 with the result that the spatial domain method is more popular and mostly used by more number of authors in comparison to frequency domain (as shown in Figure 4). Yet, there is a lot of scope and opportunities to steganalysis and develop more and effective methods for

steganography [7].

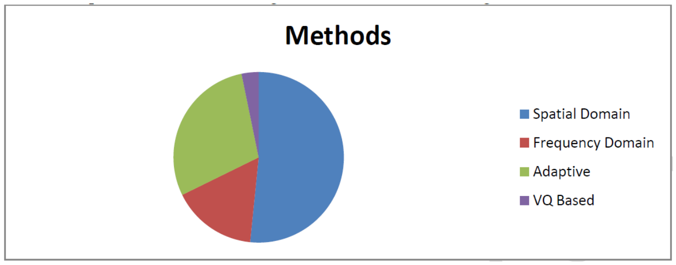

Figure 4: Stego Methods

The focus of this paper will be on spatial domain the most popular technique. Moreover, in spatial domain, the steganographer modifies the secret data and the cover image, which involves re-encoding the least significance bits (LSBs) in the carrier image, to the human eye these changes in the image value of the LSB are imperceptible [9] the strength of the detection approach is that it will preform on all types of images.

2.1.1 Least Significant Bits LSB

It is the steganography approach of embedding data at the least significant bit (LSB) in the cover image.

Least Significant Bits (LSB) considered to be one of the simplest approaches of embedding data in a cover image. Yet, it's one of the most difficult approaches to be beaten. This technique embeds the bits of the secret data directly into the least significant bit plane of the cover image [5].

The changes will be only made on three bits, on average, only half of the bits in an image will need to be modified to hide a secret data using the maximal cover size. In fact, The result of these changes are too small to be recognized by the human visual system (HVS), so the message is effectively hidden [1].

3. STEGAANALYSIS

Steganalysis is the main step in the steganography detecting technique to discover the hidden messages. It's the way of identifying the suspected medium , determine whether or not they have an embedded data into it, and, if possible, recovering that data. in other words 'Steganalysis is the science of attacking steganography in a battle that never ends' [5].

Steganlysis can be challenging sometimes more than cryptanalysis, to explain, the steganalyst have first to identify the suspected cover file, then locate the hidden message where sometimes it can be scattered in to more than one place in the cover file, finally, most likely the secret message will be encrypted as an extra lever of hiding it. Whereas the cryptanalyst main mission usually will be to decrypt the encrypted message.

Stegnalysis can be done according to different attacks:

1. **If the Steganography attack is known to the stegnalysis**: the cover file, the hidden message and the Steganography tool (algorithm) are all known to the steganlysis, then its shouldn't be difficult to identify and locate the hidden message.
2. **Only the original file (before embedding the message) and the Cover file are known to stegnalysis**: then the mission will be to compare the two files and identify the hidden message according to the pattern differences that are detected between the two files.
3. **If the secret message only is known to the stegnalysis**: then the mission will be looking for a known pattern in all the files , and that may be very difficult to achieve.
4. **Only the cover file is known to the stegnalysis**: similarly to the previous point it will be challenging to identify the hidden message location since it may be scattered to more than one place nor to understand it since it might be encrypted.

However, image processing are usually the main technique used in building steganalysis programs to study different image manipulations such as translating , filtering cropping and rotation. Also by examining the cover image structure for first order statistics (histograms) or second order statistics (correlations between pixels, distance, direction). JPEG double compression and the distribution of DCT (discrete cosine transform) coefficients can give hints on the use of DCT-based image steganography [5]

The focus of this paper will be under a new kind of attack where the type of the hidden message only is known (URL) and the used technique in hiding it (LSB) in all types and Images.

3.1 URL in Image Steganography

Embedding data in images is not a new technique. However, improving this method in image Steganography is getting better and more sophisticated by the day, one of these recent improvement is embedding a URL (Uniform Resource Locator) in the image least significant bits either to direct the receiver to a web page that include the secret data or the embedded URL can belong to a virus that will harm the image receiver either by destroying data or stealing data. The main reason behind embedding URL in an image instead of the whole secret data is that the URL will occupied much less space in the carrier[10] and that will help in hiding it and prevent the chance of being corrupted by the image manipulations.

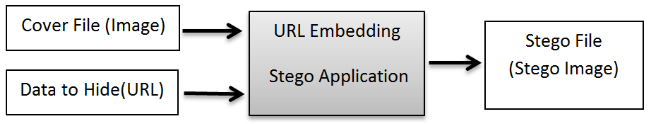

Figure 5: URL Stego Embedding Scenario

Regarding viruses, Dell SecureWorks Counter Threat UnitâĎć published on june 2015 a full analysis of the Stegoloader malware which a Stealthy Information Stealer and they explain that malware authors are evolving their techniques to evade network and host-based detection mechanisms. Stegoloader could represent an emerging trend in malware by the use of digital image steganography to hide malicious code. In fact,(Stegoloader has a modular design and uses digital steganography to hide its main module's code inside a Portable Network Graphics (PNG) image downloaded from a legitimate website)[11].

Furthermore, another analysis by the previous source Dell SecureWorks on another malware (Lurk Downloader)whereas this malware specifically embed a URLs into an image file by inconspicuously manipulating individual pixels.(The resulting image contains additional data that is virtually invisible to an observer. Lurk's primary purpose is to download and execute secondary malware payloads)[12].

4. THE PROBLEM

We are dealing with an URL hidden inside an image. As described in section section 2, any malicious code can be embedded by using least significant bit. Modifying LSB means modifying the colour by using least significant bits of an image. We have different type of color formats such as 8 bits, 24 bits etc. They have both color and grey scale. 8 bits colour means each pixel can have any of 256 (2^8) color. The same calculation is applicable 8 bits grey scale or 24 bit colors. So modifying the least significant bit in an array of huge combination of colors does not make much difference in human eye. This makes the use of LSB URL attack

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

For the simplicity of example, lets see how first character 'h' of our example url http://exampleattack.com can be added by using least significant bits of an image. The ASCII value for 'h' is decimal 104 and binary 01101000

Before LSB insertion lets assmume that 8 consecutive bytes of an images is below.
10000010 10100110 11110101 10110101 10110011 10010111 10000100 10110001

After inserting 'h' (01101000) in least significant bits the result iss below.
1000001**0** 1010011**1** 1111010**1** 1011010**0** 1011001**1** 1001011**0** 1000010**0** 1011000**0**
In this way by using more significant bit of images we can embed the rest of the characters of the intended URL.

5. URL DETECTION ALGORITHM

We are going to present an Algorithm overview in 5.1 and detail Pseocode in 5.2 to detect a hidden URL from the least signcant bits of an image. The detail complexity analyses is done in 5.3

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

Table 1: List of Top-Level Domains by the ICANN - for full list please refer to [13]

5.1 Algorithm Overview

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

Step 2: Create an array calleded BITMAP[], from an image taking each bit in 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 throuth the LSBCHARARRAY[], find out possible hidden url forming by http or https, www, domain name and top level domain(TLD).

5.2 The Algorithm in Pseodocode

The 4 steps given in 5.1 are presented here with detail pseodocode so that users can convert any any programming language easily with little efforts.

```
procedure FindURLInImage
    CREATE a sorted indexed array DOMAIN[] from the official top level domain list
    CREATE a BITMAP[] array from the image taking each bit
    *Comment: Loop through the BITMAP[] and create an array LSBARRAY[] with the least significant bits
    Integer i, j
    i=0
    j=0
    for i = 0 to BITMAP[] do
        if (i != 0) AND (i+1) MOD 8 = 0 then
            return LSBARRAY[j++] = BITMAP[i]
        end if
    end for
    *Comment: Loop through LSBARRAY[] and convert to a LSBCHARARRAY[] character array
    i=0
    j=0
    String t=""
    for i = 0 to LSBARRAY[] do t = STRING((t) + LSBARRAY[i])
        if (i!=0) and (i+1) MOD 8 = 0 then
            return LSBCHARARRAY[j++] = ConvertToCharacter(t)
            t = ""
        end if
    end for
    *Comment: Loop through the LSBCHARARRAY[] to detect URL by using the DOMAIN[] array Integer temp, l,s
    ' Initialize i and s outrside the loop
    i=0
    s=0
    Boolean httpOrHttpsExists
    Boolean wwwExists
    Boolean urlFound
    String URL = ""
    String OutPutURLArray[]
    for i=0 to LSBCHARARRAY[] do
        ' Initializeat start of loop
        httpOrHttpsExists = False
        wwwExists = False
        urlFound = False
        URL = ””
        J=0
        t=””
        temp = 0
        if LSBCHARARRAY[i] = ”:” then
            *Comment:Check Possibility of having an http:\\
            t = ConvertToString(LSBCHARARRAY[i-4] to LSBCHARARRAY[i+2])
            if LowerCase(t) = ”http:\\” then
                httpOrHttpsExists = True
                temp = i + 3
                URL= ”http:\\”
            end if
            t = ””
            if httpOrHttpsExists = False then
                *Comment:Possibility of having an https:\\
                t = ConvertToString(LSBCHARARRAY[i-5] to LSBCHARARRAY[i+2])
                if LowerCase(t) = ”https:
” then
                    httpOrHttpsExists = True
                    temp = i + 3
                    URL= ”https:\\”
                end if
            end if
            t = ””
            t = ConvertToString(LSBCHARARRAY[i+3] to LSBCHARARRAY[i+6])
            if LowerCase(t) = ”www.” then
                temp = temp+ i + 5
                wwwExists = True
                URL= Concat(URL,”www.”)
            end if
        end if
        if httpOrHttpsExists = False Or wwwExists = False) AND LSBCHARARRAY[i] = ”.” then
            *Comment: Check for . to find www because at this point we know that http or www trap inside the condition for ”:” failed.
            t = ””
            t = ConvertToString(LSBCHARARRAY[i-3] to LSBCHARARRAY[i])
            if LowerCase(t) = ”www.” then
                *Comment: jump the i to the new position and save in a temmorary variable
                temp = i + 5
                wwwExists = True
                URL= ”www.”
            end if
        end if
        if httpOrHttpsExists = True Or wwwExists = True then
            *Comment: Assign the position of i to find the URL
            i = temp
            t = ””
            ’At his point the existence of http of www is found. Now look for the rest of the url
            for j = i to LSBCHARARRAY[] do
                if LSBCHARARRAY[j] = ”.” then
```

Algorithm 1 Procedure FindURLInImage

```
                    URL = Concat(URL, ConvertToString(LSBCHARARRAY[j-i+1] to LSBCHARARRAY[j]))
                    i = j + 1
                    urlFound = True
                    Exit FOR
                end if
            end for
            if urlFound then
                urlFound = False
                for j = i to LSBCHARARRAY[] do
                    t = Concat(t,LSBCHARARRAY[j])
                    *Comment: Now check t in sorted top level domain list DOMAIN
                    if t EXISTS in DOMAIN[] then
                        URL = Concat(URL, ConvertToString(LSBCHARARRAY[j-i+1] to LSBCHARARRAY[j]))
                        urlFound = True
                        *Comment: Reinitialize the value of i for the next iteration
                        i = j + 1
                        EXIT FOR
                    end if
                end for
            end if
        end if
        if urlFound then
            *Comment: It is possible to have multiple URL in different position
            OutPutURLArray[s] = URL
            s = s + 1
        end if
    end for
    if s > 0 THEN then
        *Comment: URL has been found and OutPutURLArray[] contains the urls
        return OutPutURLArray[]
    end if
end procedure
```

5.3 Complexity analyses

5.3.1 Step 1 (Create a sorted list from the static official top level domain):

Space complexity: We have a known TLD list [13] . So in preprocessing stage, we create an indexed array, DOMIAN[] considering each TLD as a string. Space complexity is linear to the size of all characters plus the index of each string posotion in a sorted order. Also we create 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 in the whole sorted list, we just can look in the block starts with ’c’. The overall space complexity for the sorted list is O(M) +O(t) + O(i) where M is the total number of characters, t is the index on each TLD string which is limited to the official static list.

Time complexity: This can be step of computation can be preprocessing, so complexity is not a major issue. However it is possible to build up sorted list by radix sort [14] 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

5.3.2 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 single iteration the array is built.

5.3.3 Step 3(Make a character areay 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 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 implementation

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

5.3.4 Step 4 (Loop throuth the array, find out possible hidden url forming by http or https, www, domain name and top level domain(TLD))

Space complexity: The space complexity holds the linerity here with O(n) where n is the number of characters in the array.

Time complexity: This is a loop through the characters array. Finding first 3 parts of an URL (http/https and/or www, domaion name) are done in one go in the single loop. There are inside loops used to find the position and calculation purpose for http, https and www. The actual counter of 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 characters array are traversed just once. Looking up the 4th part, Top Level Domain (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 worst case where n is the number of items in an array. But in our case, n is narrowed down by index of each character. So the each block of searched area is n/m where m is the number characters in 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.

6. CONCLUSION

This paper described in detail from the existing research how data can be hidden in an image. Also, have dealt with hidden URL detection in the image and explained the approach as well as provided with the algorithm and Pseodocode so that it can be implemented in a programming language of user's choice with little efforts. Furthermore, the URL detection problem in an image was simplified with respect to string matching approach which can be used in other kind of string matching problem in an image. For example, users may be interested to search for malicious commands or other kind of strings hidden in the image using least significant bits (LSB) of the image. However, Implementing this algorithm as well as considering variation to detect malicious attacks by hidden data can be considered a guideline for future work.

7. REFERENCES

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

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

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

[4] N. Provos and P. Honeyman, "Hide and seek: An introduction to steganography," *Security & Privacy, IEEE*, vol. 1, no. 3, pp. 32–44, 2003.

[5] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, "Digital image steganography: Survey and analysis of current methods," *Signal processing*, vol. 90, no. 3, pp. 727–752, 2010.

[6] softonic. (2015) Xiao steganography. [Online]. Available: http://xiao-steganography.en.softonic.com/

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

[8] T. Morkel, J. H. Eloff, and M. S. Olivier, "An overview of image steganography." in *ISSA*, 2005, pp. 1–11.

[9] 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.

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

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

[12] D. S. C. T. U. Brett Stone-Gross, Ph.D. (2014) Malware analysis of the lurk downloader. [Online]. Available: http://www.secureworks.com/cyber-threat-intelligence/threats/malware-analysis-of-the-lurk-downloader/?view=Standard

[13] ICANN. (2016) 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

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