Volume 1,Issue-1,December-2011 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research paper

Available online at www.ijarcsse.com

Current Steganography Approaches: A survey

Sara Natanj*

Department of Software Engineering, Shiraz Azad University, Shiraz, Iran

sara_natanj@yahoo.com

Seyed Reza Taghizadeh

Department of Industrial Engineering, Kahje-Nassir Toosi University of Technology, Tehran, Iran

s-rezataghizadeh@sina.kntu.ac.ir

Abstract **_Steganography is the art of hiding the fact that communication is taking place. Steganographic systems can hide messages inside images or other digital objects. To a casual observer inspecting these images, the messages are invisible. In this paper we have first explained concept, and also the history of steganography. After that we have mentioned its basis; and as the main part of the paper, we have cited and explained different methods of steganography. We have also tried to mention advantages and disadvantages of them, and also the way each algorithm works.**

***Keywords*: Steganography, embedding, hiding information, Network Security**

I: INTRODUCTION:

Steganography is the art of hiding messages in such a way that only the recipient is aware of the existence of the message. This is originally derived from Greek words which mean ‘‘Covered Writing’’ .It has been used in various forms for thousands of years. For an instance, in the 5th century BC histaiacus shaved a slave’s head, tattooed a message on his skull and the slave was dispatched with the message after his hair grew back [1, 4]. It was also reported that the Nazis invented several steganographic methods during World War II such as Micro dots, and have reused invisible ink and null ciphers [2, 6, 7]. Steganography is in contrast with the cryptography concept, where the existence of the message is not disguised, but the content is obscured. By hiding information within digital images we can both disguise the existence of a secret message and obscure its content as well. The basis of steganography is related to two key factors: first, images are very large comparison with text files that we may send; and second, changes can be applied into the lower order bits of an image, and as a consequence, no perceived visual effect to the viewer will take place. Image based steganography, accompanied with strong data encryption is now a widely accepted form of secure digital communication. It is also a widely used method of digital image watermarking, which is used to prove image ownership. In the figure 1 we have shown the place of steganography in security systems, and also the different aspects of steganography. As you can see, the steganography can be applied to almost all kind of multimedia formats.

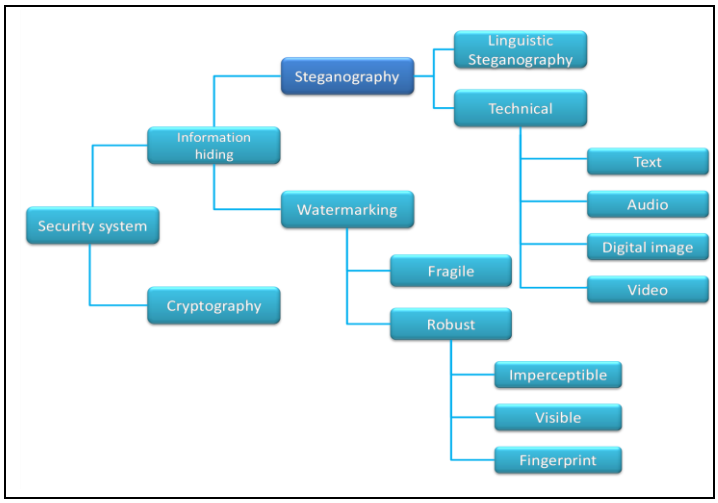

Figure 1: Steganography in security domain

If we want to have a quick comparison between steganography and watermarking, we can say that watermarking usually refers to methods that hide information in a data object so that the information is robust to modifications. That means, it is impossible to remove a watermark without degrading the quality of the data object; but steganography refers to hiding fragile information. That means modifications to the cover medium may destroy it. On the other hand, steganographic information must never be apparent to a viewer unaware of its presence, while this feature is optional in watermarking. Modern steganography should be detectable only if secret information is known, namely, a secret key. This is very similar to "Kerckhoffs' Principle" in cryptography, which holds that the security of a cryptographic system should rely only on the key material. Steganographic systems leave detectable traces within a medium's characteristics, due to their nature. This allows an eavesdropper to detect the altered media, revealing that a secret communication is going on. So, its existence is revealed, which defeats the main purpose of steganography, though the secret content is not exposed. Generally speaking, the information hiding process consists of two steps: first, identification of those bits, which can be modified without degrading the quality of the cover medium. These bits are referred to as "redundant bits"; and second, selection of a subset of the redundant bits to be replaced by those bits of secret message. In figure 2, you can see how an image would be, after being used to carry a hello word. As you can see there is no observable difference between the two images.

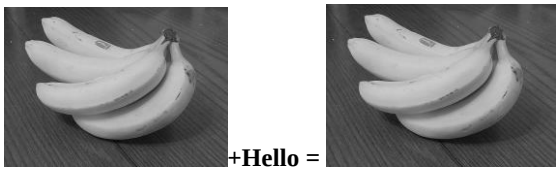

Figure 2: Left image is a plain image, while the right carries the word "Hello"

To have a more technical view of the embedding process, you can have a glance over figure 3. To make it easier to understand, we have made a brief explanation of the figure, here:

Let C denote the cover carrier, i.e., image A, and C´ the stego-image. Let K represent an optional key (a seed used to encrypt the message or to generate a pseudorandom noise which can be set to {Ø} for simplicity) and let M be the message we want to communicate, i.e., image B. Em is an acronym for embedding and Ex for Extraction. Therefore:

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

$$\therefore \; Ex(Em(c,k,m)) \approx m, \; \forall \, c \in C, k \in K, m \in M$$

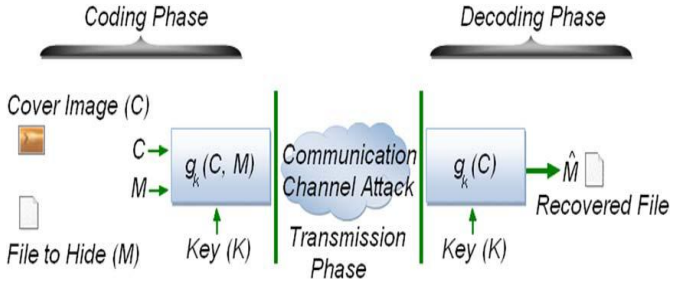

Figure 3: A theoretical view of embedding process

II: STEGANOGRAPHIC METHODS:

This part gives an overview of the most important steganographic techniques, currently used, and specially those used in JPEG images, since this is one of the most widely used formats of images. We have tried to give a summery about each method, such as its basis, and advantages.

A. LSB Substitution Method [5]

This method is the most well-known steganographic technique, used in the data hiding field. LSB stands for least-significant-bits. This method embeds the fixed-length secret bits in the same fixed length LSBs of pixels. It generally causes noticeable distortion when the number of embedded bits for each pixel exceeds three, though the technique is simple. Several adaptive methods for steganography have been proposed to reduce the distortion caused by LSBs substitution. For instance, adaptive methods vary the number of embedded bits in each pixel, and they possess better image quality than other methods using only simple LSBs substitution. However, this is achieved at the cost of a reduction in the embedding capacity. A general framework showing the

underlying concept is highlighted in Figure 4. It is reported that embedding in the 4th LSB generates more visual distortion to the cover image as the hidden information is seen as ‘‘non-natural’’[4].

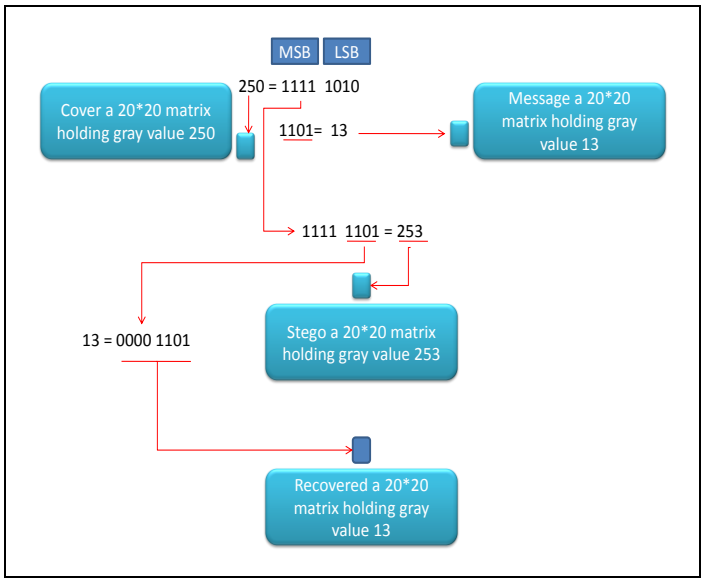

Figure 4: The effect of altering the LSBs up to 4th bit plain

A sample of this algorithm has been made by Jung and Yoo [8]. He has sampled an input image to half of its size, and then used a modified interpolation method, termed the Neighbor Mean Interpolation (NMI), to sample the result back to its original dimensions, and ready for embedding. For the embedding process the up-sampled image was divided into 4 blocks, which had no overlap, as shown in Figure 5.

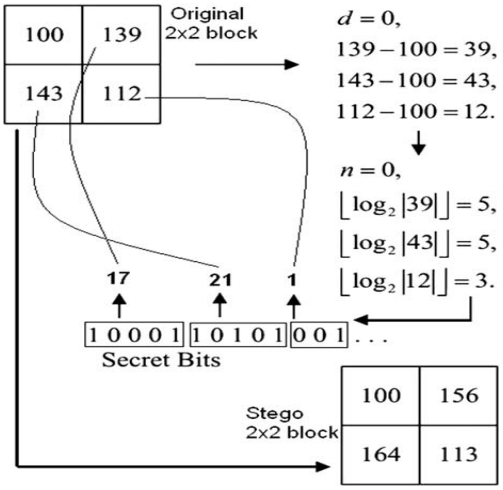

Figure 5: Jung and Yoo reported system [34].

B. Optimum Pixel Adjustment Procedure [5]:

This method reduces the distortion caused by the LSB substitution method, and referred to as “OPAP”. In OPAP method the pixel value is adjusted after the hiding secret data, to improve the quality of the stego image without disturbing the data hidden. To make a stago-image, it first substitutes a few least significant bits to be hidden. Then in the pixel, the bits before the hidden bits are adjusted suitably if necessary to give less error. The rest of the method follows an algorithmic procedure as follow:

i. Let n LSBs be substituted in each pixel.
ii. Let d= decimal value of the pixel after the substitution.
iii. d1 = decimal value of last n bits of the pixel.
iv. d2 = decimal value of n bits hidden in that pixel.
v. If (d1~d2)<=(2^n)/2

Then no adjustment should be made on the pixel

vi. Else

If (d1<d2)

d = d – 2^n

If (d1>d2)

d = d + 2^n

The final calculated d is concealed to binary and written back to pixel. The advantage of this method is its simplicity. It is also easy to retrieve the image easily, and with a higher quality than LSB substitution.

C. Inverted Pattern Approach (IP) [27]

This method tries to process the secret messages prior to embedding. There is a decision about whether invert a specified section of secret images or not. Those bits which are used to record the transformation are treated as secret keys. This method follows a specific procedure to embed the image. Suppose that the embedding message is S, the replaced string is R, and the

embedded bit string to divided to P parts. If we suppose that n-bit LSB substitution will be maid, then S and R would be n-bit length.

i. For P part in i = 1 to P
If MSE (Si,Ri) ≤ MSE(S´i,Ri)
Choose Si for embedding
Mark key (i) as logic '0'
ii. If MSE(Si,Ri) ≥ MSE(S´i,Ri)
Choose S´i for embedding
Mark key (i) as logic '1'
iii. MSE – Mean Square Error.
iv. End
v. where,
S is the data to be hidden in inverted form

In the retrieval phase, the bit will be inverted only if the key is equal to 1. After inversion of mentioned bits, the hidden data will be given. This method is also used with relative entropy [9]. Relative entropy calculates the relative entropy, instead of the mean square error for inverted pattern approach, to decide whether S suits the pixel, or S´. It measures the information discrepancy between two different sources with an optimal threshold obtained by minimizing relative entropy. To embed, it first divides the cover image into P blocks of same size, the embedding string is S, and the replaced string is R. The rest of the procedure is as follow:

i. For P part in i =1 to P
If rel.entropy(Si,Ri) ≤ rel.entropy (S´i,Ri)
Choose Si for embedding
Mark key (i) as logic '0'
If rel.entropy (Si,Ri) ≥ rel.entropy (S´i,Ri)
Choose S´i for embedding
Mark key (i) as logic '1'
ii. End
iii. where,
S is the data to be hidden
S´ is the data to be hidden in inverted form.

To retrieval procedure is the same as what was explained earlier. In probability, and information theory, the Kullback-Leibler divergence is a non-symmetric measure of the difference between two probability distributions *P* and *Q*.
It is given by,

$$D(p \parallel q) = \sum p(x) \log \frac{p(x)}{q(x)} = E_p \log \frac{p(X)}{q(X)}$$

D. Pixel Value Differencing (PVD) [32, 35, 38]

Pixel Value Differencing is able to provide a high quality stego image in spite of the high capacity of the concealed information. Here, the number of insertion bits is dependent on whether the pixel is an edge area or smooth area. If its edge area, it can be observed that the difference between the adjacent pixels is more; whereas in smooth area it is less. While human perception is less sensitive to subtle changes in edge areas of a pixel, it is more sensitive to changes in the smooth areas. Two techniques of PVD are explained here. This method hides the data in the target pixel by finding the characteristics of four pixels surrounding it.

• Select the maximum and the minimum values among the four pixel values that have already finished the embedding process. Calculate the difference value *d* between the maximum pixel value and the minimum pixel value using the following formula
"$d = g_{max} - g_{min}$ "
where,
g_{max} = max(g(x-1,y-1), g(x-1,y) ,g(x-1,y+1), g(x,y-1))
and
gmin = min(g(x-1,y-1), g(x-1,y) ,g(x-1,y+1), g(x,y-1))

By means of above equation, it is determined whether the target pixel is included in an edge area, or a smooth area .the number of bit *n*, inserted into the target pixel is determined by value d.
i. Calculate n:
If d > 3
n= $\log_2 d$ -1,
Else
n = 1 otherwise.
ii. Calculate a temporary value $t_{x,y} = b - (g_{x,y} \bmod 2^n)$ where b is the data to be hidden.

iii. Calculate t1:
If $(-(2^n -1)/2) \le t(x,y) \le (2^n -1)/2^n$
t1=t(x,y)

If $(-2^n +1) \le t(x,y) < (-2^n -1)/2^n$
$t1=t(x,y)+2^n$

If $(2^n -1)/2 < t(x,y) < 2^n$
$t1=t(x,y)-2^n$

iv. g1(x,y) = g(x,y)+t1(x,y).
g1(x,y) is the new pixel value.

E. Another PVD method:

This method also uses the concept of hiding the data using the difference between the pixel values. Unlike the previous method, this method hides the data in the difference between two adjacent pixel values. This method gives the stego images of better quality than the traditional method while maintaining a high embedding capacity. The steps for embedding are as follow:

i. Read the cover image and save the pixel values in a variable, say 'w'.

ii. Let the data to be hidden be in binary format in another variable, say ‘y‘.
iii. Leave the first row and the first column of the image.
iv. Let the pixel in which we are going to hide the data be the current pixel.
v. d1= difference (in binary) between the current pixel and its left pixel.
vi. d2= difference (in binary) between the current pixel and its top pixel.
vii. Also let l1 and l2 be the lengths of d1 and d2.
viii. Extract the data from y of lengths l1-1, l1, l1+1, l2-1, l2, l2+1 and save them in the same order in dif (i), where i=1 to 6.
ix. Now find if the d1 or d2 is nearer to any one of the dif (i). Whichever dif (i) that is nearer; let us name it as min.
x. If (min nearer to d1) && ((min~d1) < 8) (If the difference is more than 8, considerable distortion would take place in the image which is avoided.) Then adjust the current pixel value so that the difference between the current pixel and the left pixel is the data that is fetched from y, which is the data to be hidden. Save the key as ‘0‘.
xi. If (min nearer to d2) && ((min~d2) < 8) Then adjust the current pixel value so that the difference between the current pixel and the top pixel is the data that is fetched from y, which is the data to be hidden. Save the key as ‘1‘.
xii. If (min~d1)>8 && (min~d2)>8 Then skip the current pixel without embedding. Save the key as ‘ ‘.
xiii. The number of bits embedded in the pixels, which is immediately below the current pixel as follows:
a. If (the number of bits hidden<=10) then use mod10* method to hide the number of bits in the pixel below.
b. Else if (the number of bits hidden>10) then use the mod100* method to hide the number of bits in the pixel below. Our system refers to four pixels adjacent to a target pixel in the embedding process.
xiv. Repeat the above steps until all the bits in the y are hidden successfully.
xv. Then With the help of the stego image and the key file the data hidden can be extracted.

F. Adaptive steganography:

Adaptive steganography is also known as ‘‘Statistics-aware embedding’’, ‘‘Masking’’, or ‘‘Model-Based’’ [28]. This method takes statistical global features of the image before attempting to interact with its LSB coefficients. The statistics will dictate where to make the changes [29]. It is characterized by a random adaptive selection of pixels depending on the cover image and the selection of pixels in a block with large local STD (standard deviation). The latter is meant to avoid areas of uniform color (smooth areas). This behavior makes adaptive steganography seek images with existing or deliberately added noise and images that demonstrate color complexity. This method is proven to be robust with respect to compression, cropping and image processing. Edge embedding follows edge segment locations of objects in the host gray scale image in a fixed block fashion each of which has its center on an edge pixel. Whilst simple, this method is robust to many attacks and it follows that this adaptive method is also an excellent means of hiding data while maintaining a good perceptibility. Chin-Chen et al. [30], propose an adaptive technique applied to the LSB substitution method. Their idea is to exploit the correlation between neighboring pixels to estimate the degree of smoothness. They discuss the choices of having two, three, or four sided matches. The payload (embedding capacity) was high. Hioki [31], presented an adaptive method termed ‘‘A Block Complexity based Data Embedding’’ (ABCDE). Embedding is performed by replacing selected suitable pixel data of noisy blocks in an image with another noisy block obtained by converting data to be embedded. This suitability is identified by two complexity measures to properly discriminate complex blocks from simple ones; which are run-length irregularity and border noisiness this is shows in figure 6, in a good way. The hidden message is more apart of the image than being added noise [37]. The ABCDE method introduced a large embedding capacity; however, certain control parameters had to be configured manually, e.g., finding an appropriate section length for sectioning a stream of resource blocks and finding the threshold value that controls identification of complex blocks. These requirements render the method unsuitable for automatic processes. The problem which RBGC was used to solve was the complexity of the higher bit planes to tolerate little relation to the true variation of the image pixels’ intensities creating what is often called ‘‘hamming cliffs’’ [39].

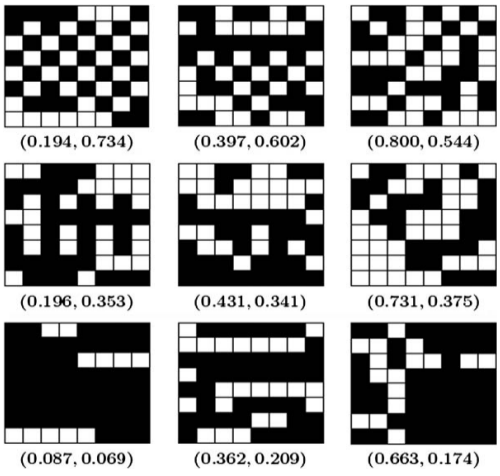

Figure 6: Blocks of various complexity values [66]

G. DCT

In this method, a transform domain technique, DCT is used to hide messages in significant areas of the cover image. Here pixels are split into 8×8 blocks. Then, all blocks are DCT transformed each block encodes exactly one secret message bit. The hiding procedure is as follow

i. The embedding process starts with selecting a block bi which will be used to code the 'i'th message bit.
ii. Let Bi = D {bi} be the DCT-transformed image block.
iii. Before the communication starts, both sender and receiver have to agree on the location of two DCT coefficients, which will be used in the embedding process. Let us denote these two indices by (u1, v1) and (u2, v2).
iv. Let m (i) be the 'i'th message bit.
v. If m(i)=0 ,
If Bi (u1, v1) > Bi (u2, v2) then
swap Bi(u1, v1) and Bi(u2, v2).
vi. Else if m(i)=1,
If Bi (u1, v1) < Bi (u2, v2) then
swap Bi(u1, v1) and Bi(u2, v2).
vii. The last step is to take inverse dct of the blocks to obtain the stego image.
viii. During the retrieval, again the stego image is split as 8X8 pixel blocks, and the blocks are dct transformed.
ix. Now, the predetermined set of two DCT coefficients are compared for all the blocks.
x. If Bi(u1, v1) > Bi(u2, v2) then the message bit=1,
xi. Else 0.

H. Hiding Streams of 1s and 0s

This method fetches the 1s or 0s present consecutively for hiding. This is different from the way the usual steganographic methods act (they fetch few bits from the secret data to be embedded). This is an innovative steganographic method where the data to be hidden is converted to binary. The number of 1s and 0s are counted and stored in the pixels of the cover image in this method. The number of 1s is stored in the odd columns of the pixel and the number of 0s is stored in the even columns. The steps of hiding the data are as follow:

i. Let the data to be hidden be in y.
ii. Find the consecutive number of 1s present in y until a '0'is encountered.
iii. Hide the cnt in the pixel using 'mod10'method.
iv. Let w(i,j) be the gray value of the pixel,
v. d=mod(w(i,j),10)~cnt, d1=mod(w(i,j),10)~(10-cnt).
vi. If(d<d1)
w(i,j)=w(i,j)-mod(w(i,j),10)+cnt
key='0'.
vii. Else
w(i,j)=w(i,j)-mod(w(i,j),10)+(10-cnt),
key='1'.
viii. If the cnt>10
Hide it in the pixel by 'mod100'method, and instead of '0'and '1', use '!' and ' '.
ix. The above steps are repeated for stream of 0s.

This method has some advantages such as simplicity of implementation, and small key sizes.

Beside these methods, there are some other methods which are basically used for JPEG images. We tried to cite some of the most famous algorithms here.

I. JSteg and JSteg-Shell

In this method, the data should be first pended with a variable size header. You can calculate the size of the length field by means of the first five bits of the header. The following bits contain the length field that expresses the size of the embedded content. JSteg-Shell is very simple. Because, it is just a user interface to JSteg, it does not encrypt the length of the embedded message. Instead it adds a signature at the end of the message. The signature is either "korejwa", "cMk4" or "cMk5". We get at least 32 bits of certainty that we guessed the right password. However, because the key size is restricted to 40 bits, it is feasible to search the whole key space. JSteg-Shell is a Windows user interface to JSteg. It has been developed by Korejwa and supports encryption and compression of the content before embedding. The data with JSteg. JSteg-Shell uses the stream cipher RC4 for encryption. However, the RC4 key space is restricted to 40 bits. When encryption is being employed, we expect the probability of embedding to be high at the beginning of the image. There should be no exception.
In retrieval phase, it should be considered that JSteg does not modify the DCT coefficients zero and one. so they are ignored in the chi-square test. We sample the DCT coefficients starting from the beginning of the image and compute the probability of embedding. This process is repeated with increasing sample size until all DCT coefficients are contained in the sample. As a performance optimization, we stop computing the probability of embedding once it falls below a certain threshold. To improve the detection accuracy, we estimate the size of the hidden content from the calculated graph and compare it with the size stored in the JSteg embedding.

J. JPHide:

The DCT coefficients are not selected continuously from beginning, JPHide is slightly more difficult to detect. The pro uses a fixed table that determines which coefficient to m next. The coefficients are selected by the table in such a way that coefficients that are likely to be numerically high are used fi pseudorandom number generator determines if coefficient skipped. The probability of skipping bits depends on the leng of the hidden message and how many bits have been embe already. JPHide not only modifies the least-significant bits o DCT coefficients, it can also switch to a mode where the sec least significant bits are modified.

In the retrieval phase we should rearrange the coefficients in that order before computing the probability of embedding, since JPHide modifies the DCT coefficients in a fixed order determined by a table. However, there are two exceptions that influence the detection. JPHide modifies the DCT coefficients −1, 0 and 1 in a special way. As a result, the modifications to these coefficients cannot be detected by the chi-square test. However, simply ignoring these coefficients still allows us to detect content embedded with JPHide. We also ignore modifications to the second-least-significant bits, which are not as frequent as modifications to the least-significant bits. Similar to JSteg, we stop computing the probability of embedding once it falls below a certain threshold.

K. Outguess:

This method chooses the DCT coefficients with a pseudo-random number generator. A user-supplied pass phrase initializes a stream cipher and a pseudo-random number generator, both based on RC4. The stream cipher is used to encrypt the content. Because the modifications are distributed randomly over the DCT coefficients, the chi-square test can not be applied on a continuously increasing sample of the image. Instead, we slide the position where we take the samples across the image.

Outguess Detection Detecting content, sometimes is complicated by the fact that the coefficients are selected pseudo randomly, there is no fixed order in which to apply the chi-square test. Instead of increasing the sample size and applying the test at a constant position, we use a constant sample size but slide the position where the samples are taken over the entire range of the image. The test starts at the beginning of the image, and the position is incremented by one percent for every application of the chi-square test. The extended test does not react to an unmodified image, but detects the embedding in some areas of the stego image. A binary search on the sample size is used to find a value for which the extended chi-square test does not show a correlation to the expected distribution derived from unrelated coefficients.

Name	Year	Encryption support	Random bit selection	Image format	Detected by	Creator
JSteg	n/a	No	No	JPEG	X^2 Test Stegdetect	Derec Upham
JSteg-Shell	n/a	RC4	No	JPEG	X^2 Test	John Korejwa
Outguess v0.13b	1994	RC4	Yes	JPEG	X^2Test (extended)	Provos & Honeyman
S_Tools	1996	IDEA,DES	No	BMP,GIF	X^2 Test	Andrew Brown
JPhide	1999	Blowfish	No	JPEG	Stegdetect	Allan LAthm
Outguess v0.2	2001	RC4	Yes	JPEG	Fridrich Algorithm	Provos & Honeyman

Table 1: A comparison of some different methods

III. CONCLUSION

In this paper we had a review over different methods of steganography. As it is shown, each method has a procedure of embedding for itself, and each one have some advantages, and also disadvantages in comparison with other methods of steganography. So it is not possible to say that a specified method is the best method and best off all. Either, it's impossible to determine the worst one. We can just compare them form different aspects, which results in determining a suitable method for a specific usage. We have also explained the way each algorithm works. It can help the reader proportionally to understand why an algorithm is better than another in a specific situation.

REFERENCES

[1] N.F. Johnson, S. Jajodia, Exploring steganography: seeing the unseen, IEEE Computer 31 (2) (1998) 26–34.

[2] J.C.Judge, Steganography: past, present, future. SANS Institute publication, <http://www.sans.org/ reading_room/ whitepapers/ stenganography/ 552.php>, 2001

[4] P. Moulin, R. Koetter, Data-hiding codes, Proceedings of the IEEE 93 (12) (2005) 2083–2126.

[5]. C.K. Chan, L.M. Chen, Hiding data in images by simple LSB substitution, Pattern Recognition 37 (3) (2004) 469–474.

[6] S. Lyu, H. Farid, Steganalysis using higher-order image statistics, IEEE Transactions on Information Forensics and Security 1 (1) (2006) 111–119.

[7] D. Kahn, The code breakers: the comprehensive history of secret communication from ancient times to the Internet, Scribner, December 5, 1996.

[8] K.H. Jung, K.Y. Yoo, Data hiding method using image interpolation, Computer Standards and Interfaces 31 (2) (2009) 465–470.

[9]. S. Katzenbeisser, F.A.P. Petitcolas, Information Hiding Techniques for Steganography and Digital Watermarking, Artech House, Norwood, MA, 2000.

[10] C. Kurak, J. McHugh, A cautionary note on image downgrading, in: Proceedings of the IEEE8th Annual Computer Security Applications Conference, 30November–4December, 1992, pp.153–159.

[11] T. L. Thomas, Alqaeda and the internet: the danger of cyber planning'', parameters, US Army War College Quarterly-Spring2003. Available from: /www.carlisle.army.mil/ usawc/ Parameters/ 03spring/ thomas.pdfS.

[12] C. Hosmer, Discovering hidden evidence, Journal of Digital Forensic Practice (1)(2006)47–56.

[13] J. C. Hernandez- Castro, I. Blasco-Lopez, J. M. Estevez-Tapiador, Steganography in games: a general methodology and its application to the game of Go, Computers and Security, Elsevier Science25 (2006) 64–71.

[14] P. Hayati, V. Potdar, E. Chang, A survey of steganographic and Steganalytic tools for the digital forensic investigator, available from: /http://debii.curtin.edu.au/ ~pedram/ images/ docs/ survey_of_steganography _and _steganalytic_tools.pdf S.

[15] W. Bender, W. Butera, D. Gruhl, R. Hwang, F. J. Paiz, S. Pogreb, Applications for data hiding, IBM Systems Journal 39 (3&4)(2000) 547–568.

[16] F. A. P. Petitcolas, Introduction to information hiding, in: S. Katzen-beisser, F. A. P. Petitcolas (Eds.), Information Hiding Techniques for Steganography and Digital Watermarking, Artech House, Inc., Norwood,2000.

[17] S. Miaou, C.Hsu, Y. Tsai, H. Chao, A secure data hiding technique with heterogeneous data-combining capability for electronic patient records, in: Proceedings of the IEEE 22nd Annual EMBS International Conference,Chicago,USA,July23–28,2000, pp. 280–283.

[18]U.C. Nirinjan, D. Anand, Watermarking medical images with patient information, in: Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Hong Kong, China, 29October–1November1998, pp.703–706.

[19]Y. Li, C. Li, C. Wei, Protection of mammograms using blind Steganography and watermarking, in: Proceedings of the IEEE International Symposium on Information Assurance and Security, 2007, pp. 496–499.

[20] D. Frith, Steganography approaches, options, and implications, Network Security 2007(8)(2007)4–7.

[21] H. Farid, A survey of image forgery detection, IEEE Signal Processing Magazine 26(2)(2009)16–25.

[22] A. Cheddad, J. Condell, K. Curran, P. Mc Kevitt, A secure and Improved self-embedding algorithm to combat digital document forgery, Signal Processing89(12)(2009)2324–2332.

[23]N. F. Johnson, S. C. Katzenbeisser, A survey of steganographic techniques, in: S. Katzenbeisser, F. A. P. Petitcolas (Eds.), Information Hiding Techniques for Steganography and Digital Watermarking, Artech House,Inc.,Norwood,2000.

[24] K. Bailey, K. Curran, An evaluation of image based steganography methods, Multimedia Tools and Applications 30(1) (2006)55–88.

[27]. C.H. Yang, Inverted pattern approach to improve image quality of information hiding by LSB substitution Pattern Recognition 41 (2008) 2674–2683

[28] P. Sallee, Model-based steganography, in: Proceedings of the Second International Workshop on Digital Watermarking, Seoul, Korea, October 20–22, 2003, Lecture Notes in Computer Science, vol. 2939, pp. 254–260.

[29] M. Kharrazi, H.T. Sencar, N. Memon, Performance study of common image steganography and steganalysis techniques, Journal of Electrical Imaging 15 (4) (2006) 1–16.

[30] C.C. Chang, P. Tsai, M.H. Lin, An adaptive steganography for index-based images using codeword grouping, Advances in Multimedia Information Processing-PCM, Springer, vol. 3333, 2004, pp. 731–738.

[31] H. Hioki, A data embedding method using BPCS principle with new complexity measures, in: Proceedings of Pacific Rim Workshop on Digital Steganography, July 2002, pp. 30–47.

[32]. C.C. Thien, J.C. Lin, A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function, Pattern Recognition 36 (11) (2003) 2875–2881.

[35]. C.M. Wang, N.I. Wu, C.S. Tsai, M.S. Hwang, A high quality steganography method with pixel-value differencing and modulus function, J. Syst. Software 81 (1) (2008) 150–158.

[37] K.B. Raja, S. Sindhu, T.D. Mahalakshmi, S. Akshatha, B.K. Nithin, M. Sarvajith, K.R. Venugopal, L.M. Patnaik, Robust image adaptive steganography using integer wavelets, in: Proceedings of the Third International Conference on Communication Systems Software and Middleware and Workshops, COMSWARE'08, 6–10 January 2008, pp. 614–621.

[38]. Young-Ran Park, Hyun-Ho Kang, Sang-Uk Shin, and Ki-Ryong Kwon, An Image Steganography Using Pixel Characteristics Y. Hao et al. (Eds.): CIS 2005, Part II, Springer-Verlag Berlin Heidelberg LNAI 3802, (2005) 581– 588.

[39] Y. Srinivasan, High capacity data hiding system using BPCS steganography, Master Dissertation, Texas Tech. University, USA, December 2003,pp.8,availablefrom: /http://etd.lib.ttu.edu/ theses/available/etd-06272008-31295018922590/unrestricted/ 31295018922590.pdfS.

[40] A. Cheddad, J. Condell, K. Curran, P. Mc Kevitt, A skin tone detection algorithm for an adaptive approach to steganography, Signal Processing 89 (12) (2009) 2465–2478.

