IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 9, Issue 2, Ver. II (Mar - Apr. 2014), PP 41-49
www.iosrjournals.org

Implementation of Segment Compression Steganographic Algorithm

[1]Shilpa S. Gaikwad, [2]Maruti B. Zalte
[1,2]*Department of Electronics and Telecommunication K.J.Somaiya College of Engineering Mumbai-77, India*

Abstract: *Steganography is the technique of hiding confidential information within any media. Using steganography, information can be hidden in different embedding mediums, known as carriers. These carriers can be images, audio files, video files, and text files. The focus in this paper is on the use of an image file as a carrier, a new steganographic technique for concealing digital images: the Segment Compression Steganographic*
Algorithm (SCSA) which is based on the Karhunen-Loève
Transform (KLT) is presented. A detailed presentation of the component parts of the algorithm follows, accompanied by quantitative analyses of parameters of interest. In addition, we make a few suggestions regarding possible further refinements of the SCSA.
Index Terms: *Steganography, Least Significant Bit, Secret Information*

I. INTRODUCTION

In Segment Compression Steganographic Algorithm the input data are first compressed using the KLT in order to achieve a higher concealing capacity, and then hidden in the least significant bits of the carrier object, which is represented in the RGB spatial domain. By combining the two procedures, we are aiming at three different research directions: increasing the capacity for concealing large messages, attaining a high quality stego object so that it is almost imperceptibly different from the carrier object and improving the execution time of the algorithm's implementation by concurrently processing different image segments (blocks) on a multi-core microprocessor. The final purpose for creating this algorithm is to implement it on yet to be released multi-core architecture mobile devices (specifically mobile phones).

The Karhunen-Loève Transform, also known as the Hotelling Transform or Eigenvector Transform allows an optimal compression, superior for instance to the one achieved by the popular Discrete Cosine Transform (DCT), the latter being in fact just an approximation of the KLT [3].The KLT completely decorrelates the input signals and is able to reallocate their energy in just a few components. KLT's greatest disadvantage with respect to other linear transforms is that it requires a great amount of processing on sometimes large sets of data. Because of this, practical implementations of the KLT algorithm require important computational resources and are lengthy in terms of execution time [1]. We plan to overcome this disadvantage by dividing a digital image into blocks (segments), thereby significantly reducing the time costs.

II. QUALITY METRICS OF STEGANOGRAHIC ALGORITHM

In order to evaluate the performance of Segment Compression Steganographic Algorithm, we took different parameters like compression rate, hiding time, recovery time, carrier error, message error, amount of data which can be embedded in carrier, etc.

The steganography algorithm alters the carrier image by embedding information pertaining to the secret message. We can calculate the difference (alteration rate) between the original carrier image *(C)* and the processed image, which we will henceforth call the stego image *(S)*. This value is the *Carrier Error*.

Also, because of the KLT compression and of subsequent processing, the message recovered *(R)* from the stego image will not be identical to the original hidden message*(M)*. The difference between *M* and *R* is the *Message Error*.

We find that the message error increases with segment size, while the carrier error decreases. Thus, we must make a compromise when it comes to choosing the segment size. If we want a carrier alteration as imperceptible as possible, than a larger segment size is indicated, but if we are more interested in the quality of the recovered message, than we should aim for a smaller segment size.

III. SCSA – step by step description

The Segment Compression Steganographic Algorithm, like any other steganographic algorithm, is composed of two perfectly mirrored parts: obtaining the steganographied image (stego), which takes place at the sender level, and recovering the payload, which takes place at the receiver level.

A. Secret Message Representation

We consider the secret message represented as a *m*× *n* RGB matrix of pixels, where *m* is the height of the image and *n* the width.

$$M = \begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{m,1} & P_{m,2} & \cdots & P_{m,n} \end{bmatrix}$$

In order to achieve a suitable representation for processing, we separate the pixels' three distinct pieces of information (Red level, Blue level and Green level), thus obtaining a (3*m*) × *n* matrix we will call *M* '. According to the RGB color scheme, *R i, j, G i, j* and *Bi, j* are all integers in the range (0,255) and correspond to the pixel *Pi, j* .

$$M' = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & \cdots & R_{1,n-1} & R_{1,n} \\ G_{1,1} & G_{1,2} & G_{1,3} & \cdots & G_{1,n-1} & G_{1,n} \\ B_{1,1} & B_{1,2} & B_{1,3} & \cdots & B_{1,n-1} & B_{1,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ R_{m,1} & R_{m,2} & R_{m,3} & \cdots & R_{m,n-1} & R_{m,n} \\ G_{m,1} & G_{m,2} & G_{m,3} & \cdots & G_{m,n-1} & G_{m,n} \\ B_{m,1} & B_{m,2} & B_{m,3} & \cdots & B_{m,n-1} & B_{m,n} \end{bmatrix}$$

B. Message Segmentation

Segmentation is the key component of the algorithm. Through segmentation, we gain both in terms of concealing capacity and execution time. We use the term *segment of size s* to refer to a submatrix of size *s*× *n* of the original pixel matrix *M*.. This segment is essentially composed of *s* contiguous lines (rows) of the segmented image. The segment corresponds to a (3· *s*) ×*n* submatrix of *M* ', since each pixel contains three independent color levels (R, G, B), which are split into three different rows in matrix *M* '.

Following our experimental tests, we have concluded that applying the KLT on image segments instead of applying the same transform on the whole, unsegmented image, yields far better results in terms of execution time and compression rates. We have experimented with different image sets and different segment sizes and the results show that the optimum segment size is situated somewhere between 5 and 8 rows (lines).

In Figure 1, we can see how the compression execution time depends on the segment size. The differences between execution times are small, but have a clear tendency. The smaller the size of the segment, the faster the compression algorithm executes. These results have a much greater significance when confronted with the execution time of the KLT on the unsegmented images. By segmenting the secret message image, we have made the compression roughly 100 times faster.

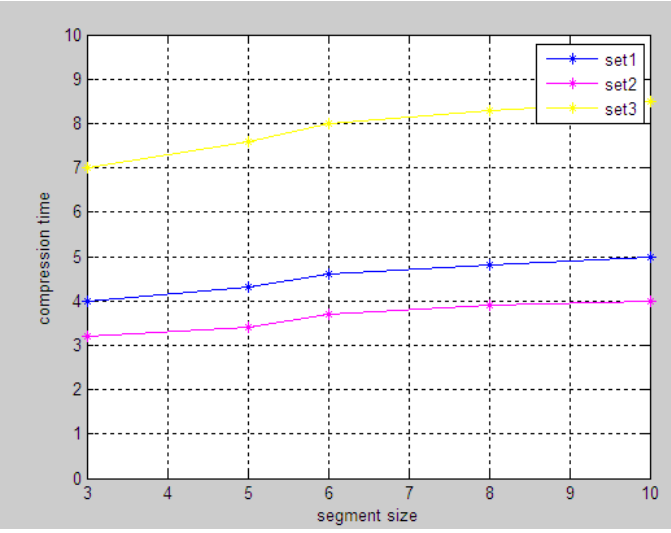


Figure 1: Compression Time Vs Segment size

The compression rate largely depends on the image color composition, but we can adjust this rate by changing the segment size. Figure 2 implies that a larger segment size accounts for a better compression rate. However, we can observe that there is a limit to the improvement in compression rate somewhere around size 10. From there on, the compression rate will start to degrade. Consequently, it is of no surprise that the compression rate for unsegmented images is so poor that the compressed message will not "fit" inside the carrier.

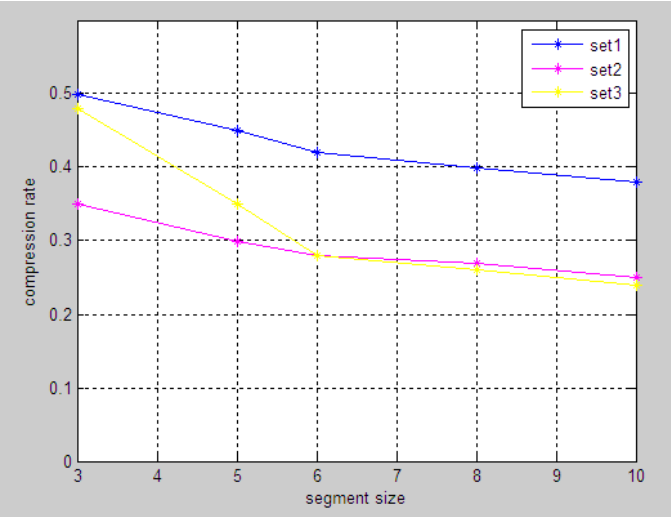


Figure 2: Compression rate Vs Segment size

C. KLT COMPRESSION

Let us consider that we have divided the image into segments of size s. The matrix M^* corresponds to the first of these segments.

$$M^* = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,n-1} & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,n-1} & x_{2,n} \\ x_{3,1} & x_{3,2} & x_{3,3} & \cdots & x_{3,n-1} & x_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{3s-2,1} & x_{3s-2,2} & x_{3s-2,3} & \cdots & x_{3s-2,n-1} & x_{3s-2,n} \\ x_{3s-1,1} & x_{3s-1,2} & x_{3s-1,3} & \cdots & x_{3s-1,n-1} & x_{3s-1,n} \\ x_{3s,1} & x_{3s,2} & x_{3s,3} & \cdots & x_{3s,n-1} & x_{3s,n} \end{bmatrix},$$

where $x_{3i-2,j} = R_{i,j}$, $x_{3i-1,j} = G_{i,j}$ and $x_{3i,j} = B_{i,j}$ $\forall$ $i \in \overline{1,s}$ and $j \in \overline{1,n}$.

From a probability and statistics point of view, we can treat each column vector *x, j* of the matrix M^* as a ample of a random n-dimensional variable *X*. We first calculate the sample vector mean ,

$$m_x = \frac{1}{n}\sum_{j=1}^{n} x_{,j}$$

And then we use this result to obtain the sample (3*s*)□(3*s*) covariance matrix

$$\Sigma_x = \frac{1}{n-1}\sum_{j=1}^{n}\left(x_{,j} - m_x\right)\left(x_{,j} - m_x\right)^{\mathrm{T}}. \text{ [9]}$$

The next step is to calculate the eigenvalue λ_i and eigenvectors $v_i \in \mathrm{R}^{3s\times 1}$ of the covariance matrix. We use the *Jacobi eigenvalue algorithm* for this purpose. Since the covariance matrix is obviously symmetric

($\sum_x = \sum_x^T$) it has an orthonormal basis of eigenvectors. These eigenvectors constitute a (3*s*)×(3*s*) orthogonal matrix $V = [v_1\ v_2 \text{----}\ v_{3s}\ ]$, which has the property $V.V^T = V^T.V = I_{3s}$[10].

It holds that $V^T \Sigma_x V = \Lambda$, where Λ is the diagonal matrix *diag* ($\lambda_1, \lambda_2, ..., \lambda 3s$) [10]. The distribution of the processed matrix's M^* energy (information) among the eigenvectors is indicated by the eigenvalue. The value of each eigenvalue is proportional to the quantity of energy stored by the corresponding eigenvector.

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_{3s} \end{bmatrix}$$

To practically use the aforementioned property, we rearrange the eigenvectors of matrix *V* so that the first eigenvector corresponds to the largest eigenvalue; the second eigenvector corresponds to the second largest eigenvalue and so on. After that, suppose we want to compress the data so that we retain only 99% of the total energy (information).

We retain only the *k* most significant eigenvectors, amounting to 99% (in our example) of the total energy of M^*. These eigenvectors form the reduced eigenvector matrix $V^* = [v_1\ v_2\ \text{---}\ v_k]$ The final step of the compression algorithm is to obtain the projection matrix:

$$Pj = (V^*)^T . M^* \qquad (1) \qquad (1)$$

The projection matrix is the compressed counterpart of the original segment submatrix M^*.*Pj* is a $k \times n$ matrix, while

M^* is a (3*s*) × *n*, so the compression rate is the ratio k/3s. By calculating the average of compression rates for each segment, we obtained the global compression rate, which is analysed in Figure 2 with respect to segment size.

D. Hiding the message

The compression rate discussed above does not take into account the internal representation of information on computers. The problem is that the entries of M^* are integer values in the range (0,255) , which can be stored in a single byte, whereas the entries of the projection matrix *Pj* are real numbers, which require floating-point representation (at least 4 bytes) to be stored. Thus, in terms of computer bytes, the compression rate is at least 4k/3s, so we may have no actual compression at all. To achieve the original compression rate, we will have to apply a linear transform on the *Pj* values. We used the following formula:

$$Pj|_{i,j} = (Pj_{i,j} - \min Pj) / (\max Pj - \min Pj) \cdot 255 \qquad (2),$$

$$\text{where } \min Pj = \left\lfloor \min_{i,j} Pj_{i,j} \right\rfloor (3) \text{ and } \max Pj = \left\lceil \max_{i,j} Pj_{i,j} \right\rceil (4).$$

This transformation ensures that the new values are in the interval [0,255] and thus can be rounded to a byte value.

The previous transformation produces an inevitable loss in compression quality. Fortunately, the losses are minor, as the original signal (information) is mainly stored in the eigenvectors. These eigenvectors need to be hidden together with the projection matrix inside the carrier image, in order to be able to rebuild M^*. We encounter the same problem: the entries of the eigenvector matrix V^* are real numbers, which need at least 4 bytes to be stored. This could seriously affect our compression rate, so we will resort to another transform in order to improve the compression rate, at the expense of compression quality.

Since the *Jacobi eigenvalue algorithm* ensures that the eigenvectors entries are in the interval [-1,1], we can use the following formula:

$$V^{*'}_{i,j} = V^*_{i,j} \cdot 32767 \qquad (5)$$

The new values are in the interval [-32767, 32767] and thus can be rounded to a two-byte (short) value.Now that we have minimized the amount of data to be hidden as much as possible, we can proceed to the actual hiding. We will hide the information in the least significant bits (LSB) of each byte of the carrier image. Depending on how many of these bits we use for hiding the message, we get three versions of the same algorithm: SCSA1, SCSA2 and SCSA4. The more bits we use for hiding, the more information we will be able to hide, at the expense of a greater Carrier Error.

In addition to the projection matrix and eigenvector matrix, for each segment we must hide the dimensions of the projection and eigenvector matrix (*k*, *n* and *3s*), and also min and max, defined by (3) and (4). All these data

are essential in recovering the secret message. It is true that the extra data worsen the compression rate, but insignificantly.

E. Recovering the message

The recovery process should seem straightforward since it is exactly the reverse of the hiding process. First, we extract the hidden data from the least significant bits (LSB) of the stego image. For each segment, we get the linearly processed projection and eigenvector matrix for each segment, their dimensions and min (3) and max (4).

Obviously the next step is to undo the linear transformations used on the matrices. The following formulas are the exact inverses of (1) and (5):

$$a\,Pj_{i,j} = Pj'_{i,j} \cdot (\max Pj - \min Pj) / 255 + \min Pj \qquad (6)$$

$$a\,V^*i,j = V^*'i,j / 32767 \qquad (7)$$

The resulted projection matrix and reduced eigenvector matrix are just approximations of their original counterparts, hence the leading "*a"* that suggests this fact. These matrices are combined to obtain an approximation of the initial segment submatrix *M* *:

$$aM^* = aV^* \cdot aPj \simeq V^* \cdot Pj \overset{(1)}{=} V^* \cdot \left(V^*\right)^T \cdot M^* \simeq I_{3s} \cdot M^* = M^*$$

By combining these recovered segments, we obtain an approximation of the original hidden message.

F. Results

To prove the steganographic quality of the SCSA algorithm, we will present the results achieved by applying the algorithm on three representative sets of images. For each image set, we used a different variant of SCSA.

The first image set was processed using SCSA1. SCSA1 ensures a very small carrier error (2.10455%). Consequently, the carrier image is barely discernible from the stego image. We can see that the recovered message quality is very high as well (message error = 1.706%). The hiding part of the algorithm took 3.67238 seconds on a Intel(R) core TM i5-2430M CPU with 2.4 GHz,4 GB RAM, while the recovery part took 1.6962 seconds. The compression rate (average of *k/3s*) was 0.493827.

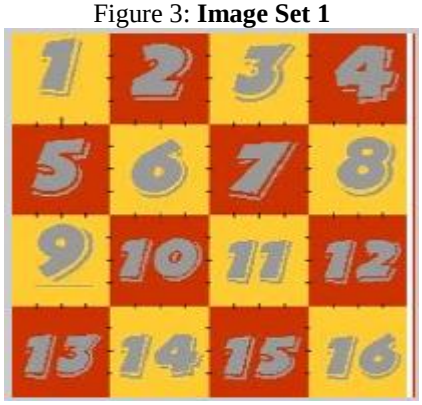
Carrier Image (600×400)

Stego Image

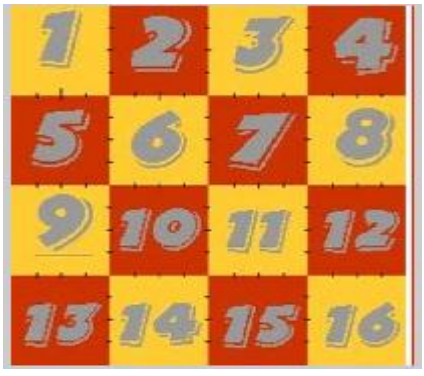
Message or Payload (200× 135)

Recovered Message

Figure 3: **Image Set 1**



Carrier Image (400 ×400)



Stego image

Message or Payload (256× 256)

Recovered Image

Fig 4: Image Set 2

The second image set was processed using SCSA2. SCSA2 yields average results in terms of steganographic quality. The carrier error is larger (2.57147%) compared to that of the first image set, but so is the size of the message we can hide. The message error is higher as well (2.5493%), but this error is only influenced by the color composition of the message itself. Using the same hardware resources, we managed to execute the algorithm in less than 5 seconds (3.2s hiding the message + 1.6s recovering the message).The compression rate achieved was also very good (0.3).

The third image set was processed using SCSA4. The carrier error is low (1.5077%). Unlike the previous two stego images, the one from image set 3 shows some clear marks of alteration. Nevertheless, this may not pose a problem when the attacker does not possess the original image for comparison. The payload is recovered almost completely (message error = 2.19197%). Since we are dealing with larger images, hiding the message took longer, about 7.939 seconds, while recovering the message took 2.9 seconds. In terms of compression rate, we achieved very good results (0.444).

The greatest advantage of SCSA4 is that it allows us to hide very large messages. Thus, it was possible to conceal(with amazing precision) a message of size 640x480 in a carrier having the same size.

Carrier image(640× 480)

Stego Image

Message or Payload(640 ×480)

Recovered Image

Fig 5: Image set 3

	SCSA1	**SCSA2**	**SCSA4**
Compression Rate	0.501253	0.333333	0.444444
Hiding Time(s)	3.82227	3.17295	7.94326
Recovery Time(s)	1.7	1.35236	2.67711
Messageerror(%)	1.05485	2.05629	1.69578
Carrier error(%)	0.718152	2.89244	1.59577

Table1: Comparison of various parameters with different variants of SCSA

IV. CONCLUSION

The strong features of the Segment Compression Steganographic Algorithm place it in a good spot for practical applications. As mentioned, the algorithm is designed and optimized for concealing digital images in other digital images. The SCSA's greatest strengths are the excellent embedding capacity provided by the KLT compression and the good visual imperceptibility provided by the LSB embedding technique [16-18]. It is also very important to restate that the SCSA has a very short execution time, given the computational complexity of image processing [19-20]. The algorithm's inherent concurrent nature recommends it for deployment on multicore platforms, for instance intelligent mobile devices with image processing capabilities.

REFERENCES

[1] S.G.Hoggar, "Mathematics of Digital Images", Cambridge University Press, 2006, ISBN-13 9780521780292

[2] Candik M., Brechlerova D., "Digital watermarking in digital images", Security Technology, 2008. ICCST 2008. 42nd Annual IEEE International Carnahan Conference on**,** 13-16 Oct. 2008, pp.43-46, ISBN: 978-1-4244- 1816-9

[3] S.G.Hoggar, "Mathematics of Digital Images", Cambridge University Press, 2006, ISBN-13 9780521780292

[4] Dafas P., Stathaki, T., " Digital image watermarking using blockbased Karhunen-Loève transform", Image and Signal Processing and Analysis, 2003, ISPA 2003, Proceedings of the 3rd International Symposium, 18-20 Sept. 2003, pp. 1072 – 1075,Vol.2, ISBN: 953-184-061-X

[5] Piva A., Bartolini F., Boccardi L., Cappellini V., De Rosa A., Barni M.,"Watermarking through color image bands decorrelation",Multimedia and Expo, 2000, ICME 2000, IEEE International Conference on**,** 30 July-2 Aug. 2000, pp. 1283 - 1286 vol.3, New York,ISBN: 0-7803-6536-4

[6] Moulin, P. Ivanovic, A. , "The zero-rate spread-spectrum watermarking game"**,** Signal Processing IEEE Transactions on**,** Apr 2003, Vol. 51, Issue: 4, pp. 1098- 1117, ISSN: 1053-587X

[7] Stanescu, D, Stratulat, M., Ciubotaru, B,. Chiciudean,D, Cioarga, R., Borca, D, "Digital Watermarking using Karhunen-Loeve transform", 4th International Symposium on Applied Computational Intelligence and Informatics, 2007. SACI '07, 18 May 2007, pp. 187-190, Timisoara Romania, ISBN:1-4244-1234X

[8] Stanescu, D, Groza, V, Stratulat, M, Borca, D, Ghergulescu, I,"Robust Watermarking with High Bit Rate", Third International Conference on Internet and Web Applications and Services, 2008, ICIW2008, 8-13 iune 2008, Athena, Greece, pp. 257-260, 2008, ISBN: 978-0-7695-3163-2

[9] Emilia Petrişor," Probabilităţi şi statistică. Aplicaţii în economie şiinginerie", Editura "Politehnica" Timişoara, 2007, ISBN 947-625-210-8

[10] G. Strang," Introduction to Linear Algebra", Wellesley-Cambridge Press, 2003, (UPT library).

[11] Daniela Sănescu, Valentin Stângaciu, Ioana Gergulescu, Mircea Stratulat, Steganography on Embedded Device, 5th International Symposium on Applied Computational Intelligence and Informatics,SACI 2009, ISBN:978-1-4244-4478-6, Timisoara, 2009, pp.313-317

[12] Boncelet C., MarvelL., Lossless Compression- Based Stegnalysis of LSB Embedded Images, 41st Annual Conference on Information Sciences and Systems, CISS'07,Baltimore, 14-16 March 2007,ISBN:1 - 4244-1037-1, pp.923-926

[13] Feng H., Effros M., On the rate-distortion performance and computational efficiency of the Karhunen-Loeve Transformfor lossy data compression, IEEE Transaction on Image Processing, feb. 2002, vol.11, Issue2, pp.113-122

[14] Ki-Hyun Jung, Kyeoung-Ju Ha, Kee- Young Yoo, Image Data Hiding Method Based on Multi-Pixel Differencing and LSB Substitution Methods, International Conference on Convergence and Hybrid

[15] Informatin Technology, ICHIT"08, Daejeon, 28-30 Aug., 2008,ISBN: 978-0-7695-3328-5, pp. 355-358

[16] Xiaolong Li, Tieyong Zeng, Bin Yang, Improvement of the Embedding Efficiency of LSB Matching by Sum and Difference Converting Set, IEEE International Conference on Multimedia,Hannover, June 23, 2008, pp.09-212

[17] W. Burger, M. Burger, Digital Image Processing, Springer, 2008,ISBN:978-1-84628-379-6

[18] Chang C-C., Chou H., Lin c-C., Colour Image-hiding scheme using human visual system, Imaging Science Journal, Oxford, UK, sept. 2006, vol. 54, nr.3, pp.152-163

[19] Eric Cole, Hiding in Plain Sight: Steganography and the Art of Convert Communicating, Wiley Publishing, Inc., Indianapolis, SUA, ISBN: 0-471-44449-9, 2009

[20] Kawaguci E , Eason R., Large Capacity Steganography, U.S.Patent no. 6,473,516, oct. 29, 2002

[21] He Junhui Tang, Shaohua Wu Tingting, On the Security of Steganographic Techniques, Congress on Image and Signal Processing, CIPS, 2008, China, 27-30, May, pp716-719, vol.5

[22] http://www.wallbank.me.uk/gallery/albums/userpics/sparrow_001_6 00x400.jpg

[23] http://lindsaywenzel.com/ColeByrd.com/Images/battle%20hymn /eagle %25205.jpg