Steganalysis of JPEG Images: An Improved Approach for Breaking the F5 Algorithm

Hatim A. Aboalsamh **Hassan I. Mathkour** **Mona F. M. Mursi** **Ghazy M.R. Assassa**

Department of Computer Science
College of Computer and Information Sciences
King Saud University

Abstract

People often transmit digital images over the internet and JPEG is one of the most common used formats. Steganography is the art and science of hiding communication; the information hiding process thus uses an image as a cover medium to embed a hidden message. Steganalysis is the inverse process of trying to identify the existence of hidden message in a cover image. In this paper, we present an enhancement to the steganalysis algorithm that successfully attacks F5 steganographic algorithm. The key idea is related to the selection of an "optimal" value of β (the probability that a non-zero AC coefficient will be modified) for the image under consideration. Rather than averaging the values of β for 64 shifting steps worked on an image, an optimal β is determined that corresponds to the shift having minimal distance E from the double compression removal step. Numerical experiments were carried out to validate the proposed enhanced algorithm and compare it against the original one. Both algorithms were tested and compared using two sets of test images. The first set uses reference test data of 20 grayscale images [1], and the second uses 432 images created by manipulating 12 images for various image parameters: two sizes (300x400 and 150x2000), six JPEG old quality factors (50, 60, 70, 80, 90, 100), and 3 message lengths (0, 1kB, 2 kB). The results suggest that the original algorithm may be used as a classifier, since it shows a good detection performance of both clean and stego test images; whereas, the proposed enhanced algorithm may be used as an estimator for the true message length for those images that have been classified by original algorithm as stego images.

Keywords

Steganalysis, Steganography, F5 algorithm, Discrete Cosine Transforms DCT, Matrix encoding, JPEG images.

1. Introduction

Steganography is the art of hiding messages inside unsuspicious medium. It's a new method to establish a secure communication between two parties. The purpose of steganography is to hide the existence of a message from a third party [2]. Steganography is a word that comes from Greeks which means 'covered writing'. It was used at least before 2,500 years, when Demaratus hide a secret message using a wax as a cover, in order to warn Sparta of a near attack on Greece [4]. In the past, null cipher is an example of the use of steganography. Null ciphers are a way to hide a message within another message using a simple, not complicated algorithm.

In the internet, spam mails are a sort of steganography medium where the body of the message is a rich text that has no meaning to us except to the end party [2]. Document may also be used to hide information by adjusting the layout, like the position of lines and words [5].

Cryptography is widely used with steganography, but they don't represent the same idea; they are two different topologies. A major drawback of cryptography is that the existence of the data is not hidden [6], whereas, steganography hides the existence of the data to be noticed [7]. Although steganography hides covert message, the fact that two parties are communicating with each other is obvious [2]. Cryptography and steganography are used for the same reason which is to establish a secure communication between two parties; cryptography hides the content of the secret message and steganography hides it's existing; besides, steganography can embed cryptography to hide the content too.

Images and multimedia components, such as video and audio files, are widely used and exchanged through the internet. Such mediums are the best cover media to hide messages inside. Secret message bits are inserted in an area of the cover file (carrier) that is not to be noticed either by eye or ear of a third party. In audio files, sounds of higher amplitude can include small echoes delays or can be masked with subtle signals [8].

2. Related Work

Reference [18] discusses existing stegnographic systems and presents research activity in detecting them via statistical steganalysis. In [19, 20] surveys are

carried out that focus on general usage of information hiding and provide an overview of detection algorithms.

Least significant bit (LSB) is the simplest form of steganography. LSB is based on inserting data in the least significant bit of pixels, which lead to a slight change on the cover image that is not noticeable to human eye. As a simple example of LSB substitution, imagine "hiding" the character 'G' across the following eight bytes of a carrier file (the LSBs are underlined):

```
10010101 00001101 11001001 10010110
00001111 11001011 10011111 00010000
```

A 'G' is represented in the American Standard Code for Information Interchange (ASCII) as the binary string *01000111*. These eight bits can be "written" to the LSB of each of the eight carrier bytes as follows:

```
1001010*0* 00001101 1100100*0* 10010110
0000111*0* 11001011 10011111 0001000*1*
```

In the sample above, note that only half of the LSBs are actually changed (shown above in underline bold italics). This actually makes some sense when one considers that we are substituting one set of zeroes and ones with another set of zeroes and ones.

LSB substitution can be used to overwrite legitimate RGB color encoding or palette pointers in GIF and BMP files, coefficients in JPEG files, and PCM levels in audio files. By overwriting the LSB, the numeric value of the byte changes very little and is least likely to be detected by the human eye or ear.

LSB substitution is a simple, albeit common, technique for steganography. Its use, however, is not necessarily as simplistic as the method sounds. Only the most naive stego software would merely overwrite every LSB with hidden data; almost all use some sort of means to randomize the actual bits in the carrier file that are modified. This is one of the factors that make stego detection so difficult. One other way to hide information in a paletted image is to alter the order of the colors in the palette or use LSB encoding on the palette colors rather than on the image data. These methods are potentially weak. However, many graphics software tools order the palette colors by frequency, luminance, or other parameters; and a randomly ordered palette stands out under statistical analysis [2]. Since there are only three LSB's for each pixel in RGB image, the total number of bits that can be hidden is only three times the total number of pixels. For example, for an image dimensions 768x512, the image has a total hiding capacity of 147,465 bytes [9]. Converting an image from a format like GIF or BMP, which reconstructs the original message exactly (lossless compression), to a JPEG, which does not reconstruct the original message exactly (lossy compression), and then back could destroy the information hidden in the LSBs. On average, only one half of the LSBs are changed [10]. LSB are used in paletted images formats like GIF by inserting bits in the palette of the image and resorted it again.

±k steganography is an improved model of LSB model. The ±K embedding for k = 1 is a general form of LSB embedding. Instead of flipping the LSB, the sender increases or decreases the pixel value by one so that its LSB matches the message bit. This light modification of the LSB embedding is significantly harder to detect because the pixel values are no longer paired. As a result, none of the existing attacks on LSB embedding can be adapted to attack ±1 embedding [11]. The major drawback of ±k steganography is that the embedding increases the number of color neighbors quit significantly even for low embedding rates [12].

In JPEG image format, message bits are inserted in the DCT (Discrete Cosine Transforms) coefficients. The image is divided into 8x8 blocks for each separate color component. The goal is to find blocks where the amount of change in the pixel values (the energy) is low. If the energy level is too high, the block is subdivided into 8x8 subblocks until the energy is low enough. Each 8x8 block (or subblock) is transformed into 64 DCT coefficients that approximate the luminance (brightness, darkness, and contrast) and chrominance (color) of that portion of the image. JPEG is generally considered to be lossy compression because the image recovered from the compressed JPEG file is a close approximation of, but not identical to, the original [2].

The F5 steganographic algorithm was introduced by the German researchers Pfitzmann and Westfeld in 2001 [16]. It's a ±k algorithm in the DCT frequency domain. The F5 algorithm embeds message bits into randomly-chosen DCT coefficients in JPEG images and employs matrix embedding that minimizes the necessary number of changes to embed a message of certain length. According to the description of the F5 algorithm version 11, the program accepts five required inputs: 1) quality factor of the stego-image; 2) input image file name of format TIFF, BMP, JPEG, or GIF; 3) output file name; 4) file containing the secret message; and 5) user password to be used as a seed for PRNG, and optionally comment to be inserted in the header. PRNG is a Pseudo-Random Number Generator that is seeded with a seed derived from a user-specified stego key or a pass phrase. In the embedding process, the message length and the number of non-zero non-DC coefficients are used to determine the best matrix embedding that minimizes the number of modifications of the cover-image. Matrix embedding has three parameters (c, n, k), where c is the number of changes per group of n coefficients, and k is the number of embedded bits. In their paper [13], the authors describe a simple matrix embedding (1, 2k–1, k) using a "hash" function that outputs k bits when applied to 2k– 1 coefficients. The embedding process starts with deriving a seed for a PRNG from the user password and generating a random walk through the DCT coefficients of the cover image [1]. The output of the PRNG is used to generate a pseudo-random walk through the image samples [14]. The PRNG is also used to encrypt the value k using a

stream cipher and embeds it in a regular manner together with the message length in the beginning of the message stream. The body of the message is embedded using matrix embedding, inserting k message bits into one group of 2k–1 coefficients by decrementing the absolute value of at most one coefficient from each group by one [1].

The OutGuess steganographic algorithm is a LSB algorithm that was proposed to counter the statistical chi-square attack. In the first pass, OutGuess embeds message bits along a random walk into the LSBs coefficients while skipping 0's and 1's. After embedding, the image is processed again using a second pass. This time, corrections are made to the coefficients to make the stego image histogram match the cover image histogram. Because the chi-square attack is based on analyzing first-order statistics of the stego image, it cannot detect messages embedded using OutGuess [15]. The embedded data always consist of two parts – a header and the message data. The header is embedded in coefficients determined by the secret shared stego key. It contains information about the message length and a session key. The embedding path in OutGuess is a function of the message length and the session key. The pseudorandom selection of coefficients proceeds sequentially through the image while the offset between selected coefficients is subjected to pseudo-random variations tuned so that OutGuess goes through the whole image during the first pass.

3. F5 Steganographic Algorithm

F5 uses AC DCT coefficients to carry out the secret message bits. The image is first decompressed to the spatial domain. Secondly, we compress the image by the user quality factor. Then, we estimate the medium capacity, C, with no embedding matrix:

$$C = h_{DCT} - h_{DCT}/64 - h(0) - 0.51h(1) \qquad eq.\ (1)$$

where h_{DCT} is the number of all DCT coefficients, h(0) is the number of AC DCT coefficients equal to zero, h(1) is the number of AC DCT coefficients with absolute value 1, $h_{DCT}/64$ is the number of DC coefficients, and 0.51h(1) is the estimated loss due to shrinkage. The parameter C, along with message length, determines the best embedding matrix. The user password is used as a seed for a random generator that specifies the random walk throw the DCT coefficients; also it used to encrypt the message size to be embedded ahead of the secret message. The embedding process begins by dividing the message into k-bit segments and tries to embed it into the next 2k-1 coefficients group. Zero's coefficients are skipped. Now we hash the current k-bits of message in the current 2k-1 non-zero coefficient group. If the value is 0, this means that the k-bit is embedded with no change. If the value is j, then we decrement the absolute value of the coefficient a_j. If the new value of a_j is not zero, then we had embedded successfully the current k-bit with only 1 change. If the new value of a_j is zero, this means that shrinkage had occurred, and then we remove a_j from the list and take on more coefficients and try the process again. This process is repeated until either the list of non-zero coefficients or the list of message bits is finished. If the coefficient list finishes before the message bit list, this means the F5 algorithm can not embed the entire message bit. F5 distributes the changes uniformly by generating a random walk from the user password. Only with the same password the receiver should extracts the secret message through the correct random walk process. F5 is probably the first steganographic algorithm that uses matrix encoding algorithm to minimize the necessary number of changes required for embedding secret messages [16].

4. F5 Steganalysis Algorithm "Original Algorithm" [1]

F5 attack is a steganalysis algorithm that estimates the true message length. It uses a baseline image close to the clean image after F5 compresses it with the user quality factor, along with the stego image. Based on the two images AC DCT histogram differences, the steganalysis algorithm estimates β, the probability that a non-zero AC coefficient will be modified. After evaluating β, we can further estimate the true secret message length. The baseline image is evaluated from the stego image directly by decompressing it to the spatial domain. Then we crop the image by 4 pixels in both directions, see Figure 1, and apply a low-pass filter to remove possible JPEG blocking artifacts.
The value of β is calculated using histograms of low frequencies [1, 2], [2,1], and [2,2]:

$$\boldsymbol{B_{kl}} = \frac{\hat{h}_{kl}(1)[H_{kl}(0)-\hat{h}_{kl}(0)]+[H_{kl}(1)-\hat{h}_{kl}(1)][\hat{h}_{kl}(2)-\hat{h}_{kl}(1)]}{\hat{h}_{kl}(1)+[\hat{h}_{kl}(2)-\hat{h}_{kl}(1)]^2} \qquad eq.\ (2)$$

where H and h^ represent stego and baseline absolute histogram respectively, β is then calculated as the average of the three low frequencies [1,2],[2,1],[2,2]:

$$\beta = (\beta_{12} + \beta_{21} + \beta_{22}) / 3 \qquad eq.\ (3)$$

There exist some images for which the algorithm can't estimate β correctly because of the double compression effect. Double compression takes place when the image sent to F5 is already a JPEG compressed image. The authors of F5 steganalysis algorithm [1] add a double correction step to their algorithm to eliminate this problem. They calculate β for fixed quantization tables $\{q_1 \ldots q_t\}$; and for each q_i, $1 \le i \le t$, they run the detection algorithm with a little change. After shifting and filtering processes, they compress the image with q_i and immediately dec**o**mpress it again after going throw the rest of the algorithm. Now they have a set of calculated β_i ,$1 \le i \le t$. For each i and for each DCT mode k,l , they calculate the $\boldsymbol{L}_2$ distance $\boldsymbol{E}^{(i)}{}_{kl}$ between H_{kl} and $\hat{h}_{kl}$ with $\beta = \beta_i$:

$$E^{(i)}_{kl} = [H_{kl}(0) - \hat{h}_{kl}(0) - \beta_i \hat{h}_{kl}(1)]^2 + \sum_j [H_{kl}(j) - (1-\beta_i)\hat{h}_{kl}(j) - \beta_i \hat{h}_{kl}(j+1)]^2 \quad eq.\ (4)$$

The final β is obtained as $\beta = \beta_t$ where:

$$t = \arg\min \sum_{kl} E_{kl}^{(i)} \quad eq.\ (5)$$

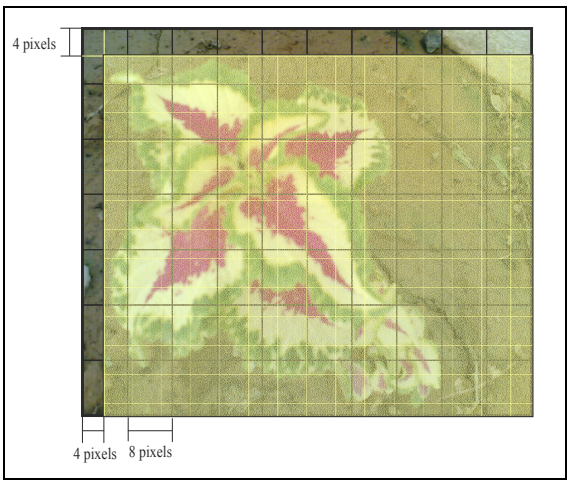

Figure 1: Image shifting example of 4 pixels in both directions.

5. Improving the F5 Algorithm

The proposed algorithm is an enhancement of the original algorithm discussed above where the value of β is calculate in a different way. The original algorithm [1], uses 64 shifts to calculate the baseline, 8 shifts in both directions, and the final value of β is calculated by averaging the 64 β values for all shift. The key idea behind the proposed algorithm is related to the selection of an optimal value of β for each image. From the numerical experiments that we carried out, it was observed that for each image there exists a value β, corresponding to a particular shift out of the 64 shifts, that is closest to the actual β than all other 63 values of β, see figure 2. This β is found to be better than the average of β's used to estimate the final β [1]. The proposed algorithm determines the optimal β from the 64 β's by the E value calculated when we chose the best β from the double compression effect removal step. Consequently, the original algorithm [1] is altered so that instead of evaluating the final β as the average of the 64 β's, it is evaluated such that the optimal β is the one whose E value is the minimum among all other 63 β's E values. The rest of the original algorithm [1] is left unchanged.

6 Validation of the proposed Improvement

Using Matlab, a tool was built based on the proposed improvement discussed above. This section discusses the validation of the tool and compares the numerical results obtained from the original and enhanced algorithms.

6.1 Test Data

To validate the proposed enhancement in algorithm [1], two data sets were used; a first set consisting of 432 test images that we created, and a second set of 20 reference test images taken from [1]. The first data set was created from 12 images taken by a NOKIA 6630 mobile camera. Each of the 12 images was resized to two different sizes, (150x200) and (300x400), thus yielding 24 resized images. Then, we applied a low-pass filter, with 3x3 kernel (with values $B_{22}=1-4e$, $B_{21} = B_{23} = B_{12} = B_{32} = e$, and $B_{ij} = 0$ otherwise), for each resized images. Each filtered image was saved using six JPEG quality factors, namely 50, 60, 70, 80, 90 and 100. Thus, we obtained 144 saved clean images (12 images x 2 sizes x 6 JPEG quality factors). For each of the 144 saved clean images, we used F5 algorithm to embed two secret messages of two sizes: 1 KB and 2 KB. This yields a total of 288 stego images, which together with the 144 clean images define the first test data set of 432 test images.

6.2 Test Tool

We developed a test tool using Matlab version 7 to automate the testing and comparing processes of the original [1] and proposed algorithms. The test tool is designed to accept as inputs the following parameters: (1) the clean image full path name in batch mode, (2) the range of shifts to manipulate, (3) the range of JPEG quality factors to experience, The tool performs the calculation and outputs the optimal estimated β value as explained above. In addition, we developed a report generator that allows outputting customizable conclusion reports of the estimated β values.

6.3 Discussion of Results

The experimental results discussed hereafter are obtained by applying the test tool using the original and the proposed improved algorithms on the two sets of test data outlined earlier in section 6.1.

Figure 2 shows a comparison of β values obtained from the first data set by the original and proposed algorithms for a 2 KB stego image of size 300x400 pixels (image #5) that was previously compressed as JPEG image using a quality factor value Q=90. The following information pertains to the image displayed in figure 2: secret message length equals 1 kB, actual (correct) β = 0.17512, average β (of the original algorithm) = 0.56605, estimated optimal β at shift 40 equals 0.26741, shift 40 corresponds to horizontal 5 pixels and vertical 8 pixels, estimated message length based on average β of the original algorithm equals 4.112 kB, estimated message length based on optimal β equals 1.3603 kB, true message length equals 1.71 kB. From figure 2, it could be observed that the proposed algorithm yields better values for β (optimal β = 0.26741) than the original algorithm (0.56605). The detection accuracy of the proposed algorithm is shown in table 1 for the first test set of 432 images and a threshold value T= 0.025. False positive and false negative are defined as wrong classification of stego and clean images respectively, whereas true positive

and true negative are defined as successful classification of stego and clean images respectively. When comparing the results of both algorithms, we observed for the first test data set that the proposed algorithm has successfully classified about 10% more stego images, and 5% less clean images than the original algorithm [1].

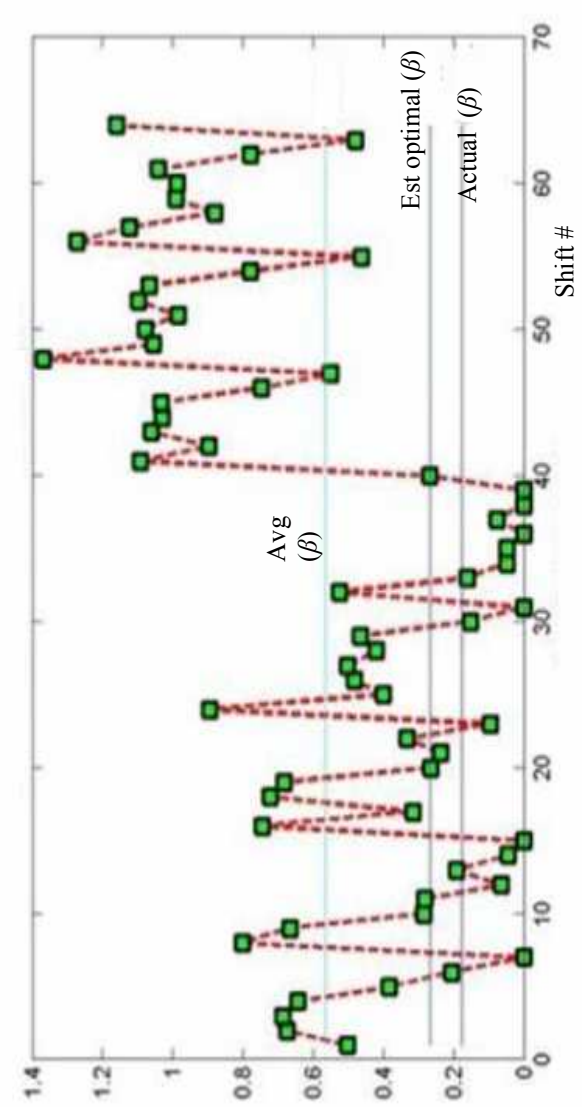

Figure 2: Comparison of β values obtained by the original and proposed algorithms: "avg(β)" refers to original algorithm, and "Est.(β)" refers to proposed algorithm.

	False positive	False negative	True positive	True negative
Image Count	53	53	235	91
Total Image Count	288	144	288	144
Percentage	18.40%	36.81%	81.60%	63.19%

Table 1: False positive, false negative, true positive, and true negative values of the proposed algorithm - 432 test images using T = 0.025.

Image #	β values		
	Actual [1]	**Estimated in Ref [1]**	**Estimated by proposed algorithm**
1	0	**0.106**	0.059
2	0.202	0.238	**0.174**
3	0	0.079	**0.054**
4	0.259	**0.273**	0.187
5	0.244	0.265	**0.236**
6	0.234	0.276	**0.250**
7	0.216	0.248	**0.207**
8	0.347	**0.409**	0.435
9	0	0.044	**0.018**
10	0	**0.07**	0.038
11	0	0.103	**0.070**
12	0.342	**0.25**	0.121
13	0.499	**0.522**	0.234
14	0	0.113	**0.086**
15	0	0.078	**0.057**
16	0.257	**0.291**	0.173
17	0	0.083	**0.038**
18	0	0.073	**0.020**
19	0.37	**0.329**	0.174
20	0.428	**0.377**	0.356

Table 2: The probability of a non-zero AC coefficient will be modified (β) actual values, estimated values in reference [1], and estimated values by the proposed algorithm.

Image #	*n* values		
	Actual [1]	**Estimated in Ref [1]**	**Estimated by proposed algorithm**
1	0	11846	**5180**
2	19845	**21937**	13359
3	0	5214	**3163**
4	20254	**19490**	11344
5	21401	**21011**	15862
6	20267	**22040**	17217
7	19675	**21176**	15192
8	24741	25873	**23610**
9	0	2570	**958**
10	0	5124	**2389**
11	0	6187	**3843**
12	23589	**15745**	6554
13	22775	**21531**	8443
14	0	8386	**5277**
15	0	4571	**3130**
16	20164	**20955**	10843
17	0	7222	**2758**
18	0	4513	**1135**
19	23930	**19342**	9111
20	24278	**19308**	17861

Table 3: The number of changed coefficients (*n*), actual values, estimated values in reference [1], and estimated values by the proposed algorithm.

In addition to the above validation, additional testing of the proposed algorithm was carried out using 20 grayscale reference test images [1] that comprise the second dataset.
Tables 2 and 3 exhibit the comparison for β and n values, respectively. Table 2 depicts, for each of the 20 images, the values of actual β, estimated β from reference [1], and the estimated β from the proposed algorithm.
Table 3 displays the same information for the values of n. Analysis of the results shown in tables 2 and 3 indicates that, (1) the proposed algorithm yields better estimation of β than the original algorithm [1] for 13 images, out of which 9 are clean images; (2) the proposed algorithm shows better estimation of n for 10 images, out of which 9 are clean images. Table 4 shows the detection accuracy of the proposed algorithm for the second dataset images; the table shows that the proposed algorithm has successfully classified all the clean images and misclassified about 9% of the stego images of the reference test data (second set).

Figure 3 shows the average absolute error of the actual β against the values of the estimated β obtained from the original and proposed algorithms for the whole test data (secret message length of 0, 1, 2 kB, and for old quality values 0f 100, 90, 80, 70, 60, and 50; a curve fitting function was applied for better presentation of the results. We carried out similar experiments for various individual old quality values of 100, 90, 80, 70, 60, and 50.

Figures 4 and 5 show the same results but for various old quality values of 80, 70 respectively; it should be noticed that we have applied a curve fitting function for better presentation of the results.

Analysis of the experimental results shown in figure 3 indicates that the proposed algorithm yields 'better' less average absolute error values of β than the original algorithm; this is practically valid for stego images with actual β values larger than 0.05. In addition, for test images that have been compressed before sending it to the F5 algorithm with old JPEG quality in the range 50 - 90, we have observed better β estimation. In particular, the proposed algorithm shows its best results for low compressed images with old quality 50. However, numerical experiments show some degradation in the results of the proposed algorithm yielding "worse' more average absolute error values of β for both clean images and images having small actual β values less than 0.05.
For threshold T = 0.025 for estimated β values, Table 5 shows that the proposed enhanced algorithm successfully classified 81.60% of the stego images, and 63.19% of the clean images, and missed classifying 18.40% of stego images and 36.81% of clean images. Table 6 shows, for the 432 test images, that the proposed enhancement yields less average absolute error values of estimated β, n, and secret message as compared with corresponding values obtained by original algorithm.

Table 7 depicts the results when applying the enhanced proposed algorithm to the 20 reference test images of [1] with the used threshold T = 0.125. The results show comparable detection performance as in [1] and better estimation of β values, reflected as less average absolute error of estimated β.

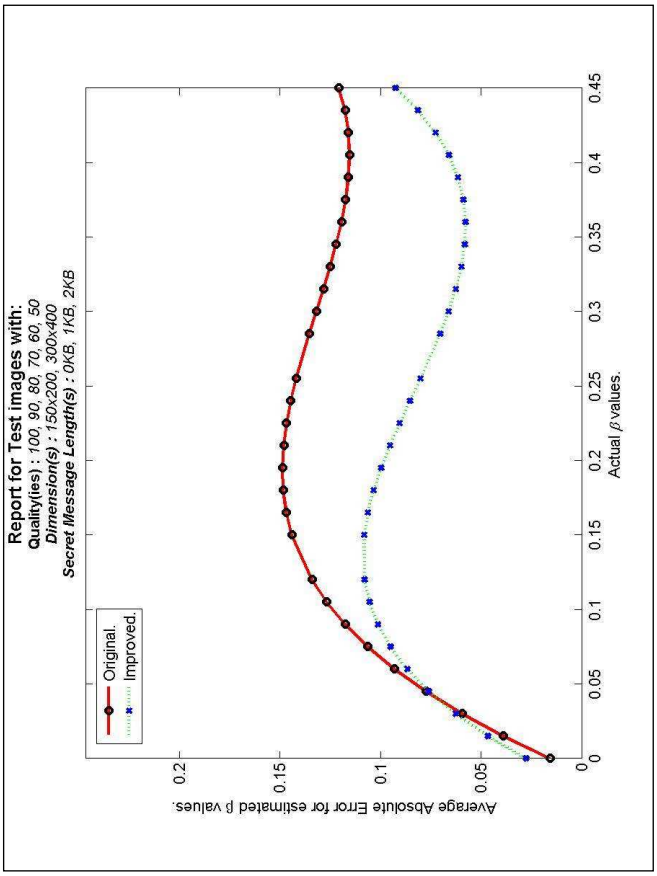

Figure 3: Average absolute error of the estimated β against the actual β obtained from the original and proposed algorithms for the whole test data.

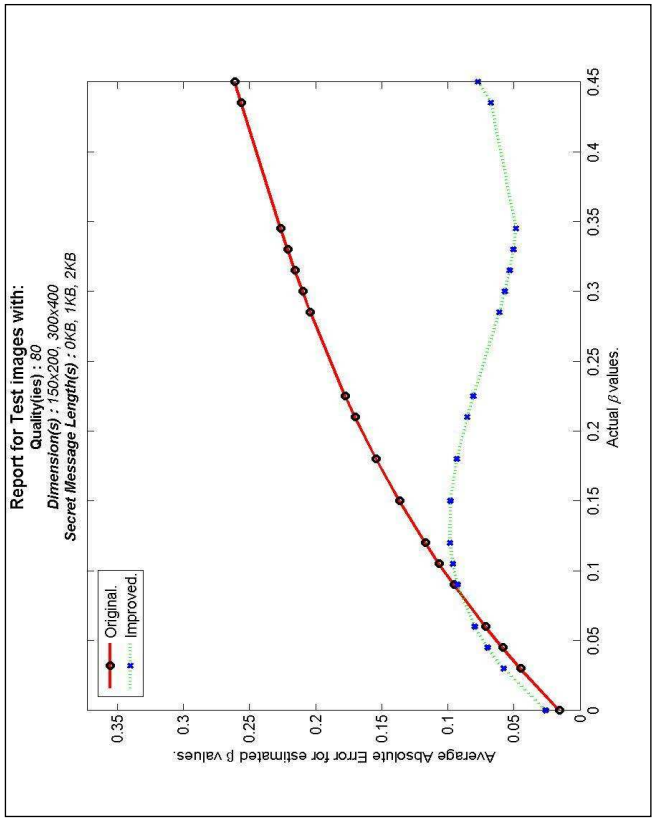

Figure 4 Average absolute error of the estimation of β against the actual β values for the original and the proposed algorithm for test data with old quality 80.

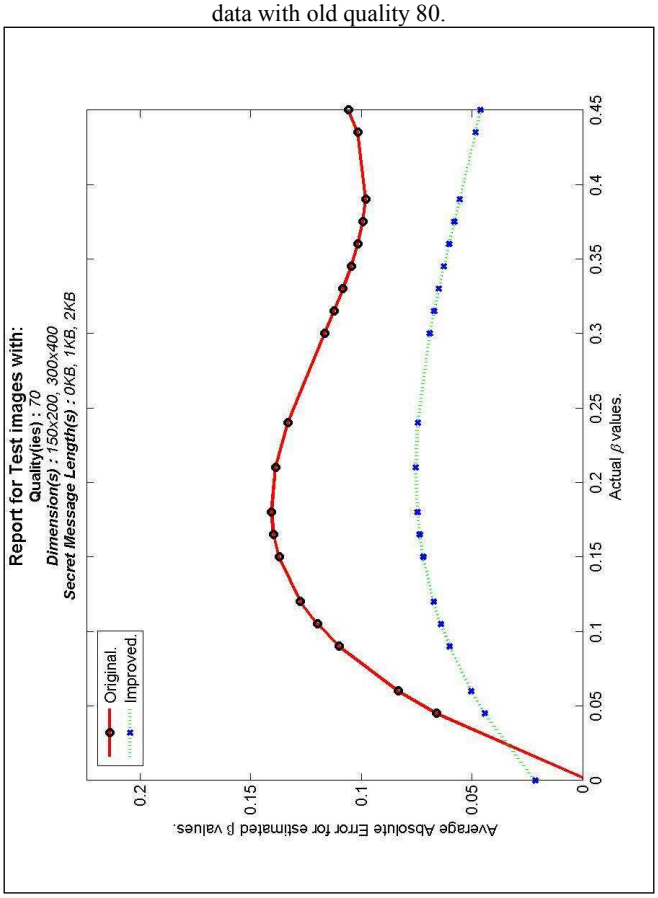

Figure 5 Average absolute error of the estimation of β against the actual β values for the original and the proposed algorithm for test data with old quality 70.

	False positive	False negative	True positive	True negative
Image Count	1	0	10	9
Total Image Count	11	9	11	9
Percentage	9.09%	0.00%	90.91%	100.00%

Table 4: False positive, false negative, true positive, and true negative values of the proposed algorithm, 20 reference test images using T = 0.025.

	False positive	False negative	True positive	True negative
Original	28.82%	31.94%	71.18%	68.06%
Proposed	18.40%	36.81%	81.60%	63.19%

Table 5: Detection statistics comparison between original and proposed algorithm results for the 432 test images using T = 0.025.

	AAE	Max (AAE)	Min (AAE)	Std (AAE)	Var (AAE)
Original β	0.084	0.528	0	0.08726	0.00761
Proposed β	0.067	0.819	0	0.08672	0.00752
Original n	3378	29080	0	4276	18281302
Proposed n	2091	17609	0	2547	6485807
Original *ML*	0.88	8.81	0.00	1.15	1.33
Proposed *ML*	0.58	5.75	0.00	0.72	0.52

Table 6 Average absolute errors (AAE) statistics of estimated β, n, and message length (*ML*) values obtained using original and proposed algorithm for the 432 test images.

Average Absolute Error	Ref [1]	Proposed algorithm
β	0.060	0.041
n	4136	4176

Table 7 Average absolute error for β and n obtained by proposed algorithm and reference [1], 20 reference test images using T = 0.025.

7. Conclusion and Future Work

We have presented an enhancement to the steganalysis algorithm [1] that attacks F5 steganographic algorithm. Rather than averaging the values of β and n for 64 shifting steps, we determine an optimal β and n values corresponding to the shift with minimum $\boldsymbol{E}$ value. The accuracy of the proposed enhancement was verified by attacking 432 images having various parameters of size, old quality factor, and message length, where 144 images out of the 432 were clean ones.

Numerical experiments indicate that the proposed algorithm yields 'better' less average absolute error values of β than the original algorithm; this is practically valid for stego images with actual β values larger than 0.05. However, results also show some degradation in the results of the proposed algorithm yielding "worse' more average absolute error values of β for both clean images and images having small actual β values less than 0.05.

Therefore, it is not recommended to use the proposed enhanced algorithm for clean images. Consequently, we suggest combining the two algorithms in a two-step approach for steganalysis. In the first step, we suggest to use the averaged β of the original algorithm [1] as a classification algorithm, since it shows good detection performance of both clean and stego test images, and then in a second step, we suggest to use the proposed algorithm to get an optimal β for the estimation of the true message length.

In future work, we plan to investigate candidate image's parameters that could be used to get the best β value. Also, the processing time is still an issue to be looked at.

References

[1] Fridrich, J., Goljan, M., and Hogea, D., "Steganalysis of JPEG Images: Breaking the F5 Algorithm", 5th Information Hiding Workshop, Noordwijkerhout, The Netherlands, 7–9 October 2002, pp. 310-323.

[2] Gary C. Kessler, "An Overview of Steganography for the Computer Forensics Examiner", Forensics Science Communications, vol. 6, no. 3, July 2004.

[3] Kefa Rabah, " Steganography-The Art of Hiding Data", Information Technology Journal 3 (3): 245-269, 2004 ISSN 1682-6027, 2004 Asian Network for Scientific Information.

[4] Hosmer, C. and Hyde, C. "Discovering Covert Digital Evidence." Digital Forensic Research Workshop (DFRWS) 2003, August 2003.

[5] Johnson, N.F. and Jajodia, S. "Exploring Steganography: Seeing the Unseen." *IEEE Computer Mag.*, February 1998.

[6] Artz, D. "Digital Steganography: Hiding data within Data." *IEEE Internet Computing*, May/June 2001.

[7] McDonald, A.D. and Kuhn, M.G. "StegFS: A Steganographic File System for Linux." In: Pfitzmann, A. (ed.). *Proc. of the Third International Workshop on Information Hiding (IH '99)*, Dresden, Germany, Sept.-Oct. 1999. *Lecture Notes in Computer Science*, Vol. 1768. New York: Springer-Verlag, 2000.

[8] Farid, H. "Detecting Steganographic Messages in Digital Images." Technical Report TR2001-412, Dartmouth College, Computer Science Dept., 2001.

[9] "Chameleon", image Steganography software developed by Mark David Gan for his thesis at STI College Bacoor, Technical Paper, April 2003.

[10] Fridrich, J., Rui Du and Long Meng, "Steganalysis of LSB Encoding in Color Images", ICME 2000, New York City, July 31-August 2, New York, USA.

[11] Fridrich, J., D. Soukal and M. Goljan, "Maximum Likelihood Estimation of Secret Message Length Embedded Using ±k Steganography in Spatial Domain", Proc. EI SPIE San Jose, CA, January 16-20, 2005.

[12] Fridrich, J., T.S. Holotyak and D. Soukal, "Stochastic Approach to Secret Message Length Estimation in ±k Embedding Steganography", Proc. EI SPIE San Jose, CA, January 16-20, 2005.

[13] Jackson, J.T., Gunsch, G.H., Claypoole, R.L., and Lamont, G.B. "Blind Steganography Detection Using a Computational Immune System: A Work in Progress." International Journal of Digital Evidence, Winter 2003.

[14] Fridrich, J., M. Goljan and D. Soukal, "Searching for the Stego Key", Proc. EI SPIE San Jose, CA, Jan 2004.

[15] Fridrich, J., Goljan, M., and Hogea, D. "Attacking the OutGuess.", In: Proc. of the ACM Workshop on Multimedia and Security 2002, Juan-les-Pins, France, Dec. 2002.

[16] Westfeld, A.: High Capacity Despite Better Steganalysis (F5–A Steganographic Algorithm). In: Moskowitz, I.S. (eds.): Information Hiding. 4th International Workshop. Lecture Notes in Computer Science, Vol.2137. Springer-Verlag, Berlin Heidelberg New York (2001) 289–302.

[17] Dokheekh, S. A., Master research in steganalysis, College of computer science and information, King Saud University, Riyadh, Kingdom of Saudi Arabia, 2006.

[18] Provos, N. and Honeyman P. "Hide and seek: An introduction to steganography", IEEE Security and Privacy, May/June 2003, pp. 32- 44.

[19] Petitcolas, F.A.P., Anderson, R.J., and Kuhn, M.G., "Information hiding: A servey", proc. IEEE, vol. 87, no. 7, 1999, pp.1062-1078.

[20] Fridrich, J. and Goljan, M., "Practical steganalysi - State of the art", Proc. SPIE Photonics Images 2002, security and watermarking of multimedia contents, vol. 4675, SPIE Press, 2002, pp. 1-13.