Impact of Steganography on JPEG File Size

Mohammad Rezaei
Security Analysis Laboratory
Tehran, Iran
rezaei@salab.ir

Saeed Montazeri Moghadam
Security Analysis Laboratory
Tehran, Iran
montazeri@salab.ir

***Abstract*—Hiding data in JPEG images is usually performed by modifying quantized DCT coefficients. This will affect the entropy coding and consequently the size of the resulting compressed data. The change in the file size might be used as a feature in steganalysis. In this paper, we investigate the impact of several well-known steganography methods on the size of JPEG file. The experiments show interesting results, where we considered several embedding payloads and quality factors. OutGuess 0.1, OutGuess 0.2, and complementary embedding methods increase the file size, while F5, nsF5, and PQ decrease it. The secure steganography methods J-UNIWARD and SI-UNIWARD almost do not change the file size.**

Keywords—steganalysis, steganography, data hiding, JPEG, file size

I. INTRODUCTION

Numerous steganography methods have been proposed for JPEG images mostly because of the popularity of JPEG format and its wide usage on various platforms [1, 2]. The main goal of steganography methods is to hide the presence of the secret data, and on the contrary, steganalysis aims at detection of the message by visual or statistical analysis [3]. Most of the JPEG steganography methods embed the secret data in the quantized *Discrete Cosine Transform* (DCT) coefficients. This can modify natural statistical properties both in the DCT domain and spatial domain of the image. Accordingly, steganalysis methods usually extract features from DCT coefficients [4-6] or spatial domain [3, 7]. Among various studies in steganography and steganalysis, very little attention has been paid to the change in file size after steganography. Discussion about the file size is usually presented only in the reversible steganography methods [8, 9] or the methods which hide data directly in the JPEG file bitstream [2]. However, all these methods are detectable by simple steganalysis methods or system attacks, which investigate the file to detect unusual and extra data in the bitstream without analyzing the image content [10].

In this paper, we analyze, theoretically and experimentally, the impact of eight well-known JPEG steganography methods on the file size. In the experiments, we consider different embedding rates and image qualities. The results show interesting properties of steganography methods, which help to employ, in the future, the file size as a feature in steganalysis methods.

The rest of this paper is organized as follows. In Section 2, we introduce JPEG compression procedure, which is necessary for theoretical analysis of a steganography method. The selected steganography methods are reviewed in Section 3. Experimental results are reported in Section 4, and the conclusions are drawn in Section 5.

II. JPEG COMPRESSION

JPEG, which is an image compression standard, is so extensive, but a small part of it called JPEG *baseline* is widely used [10, 11]. We consider this baseline and *JPEG File Interchange Format* (JFIF) in this paper. JFIF is an image file format for exchanging JPEG encoded files, which is widely used in many existing platforms and applications [12]. The first stage of the compression procedure shown in Fig. 1. is converting RGB components to YCbCr, where Y component is luminance and Cb and Cr components are color information [10]. Each component is broken down into non-overlapping 8×8 pixel blocks, and pixel values are shifted to have the range [-128,127] instead of [0, 255]. Then, 2-*D* DCT converts the data of each block into 64 DCT coefficients in the frequency domain; one DC and 63 AC coefficients [11, 13], see the example in Fig. 3. Each coefficient is quantized using its corresponding value in an 8×8 quantization table [13]. The block coefficients are then reordered by zigzag scanning to be prepared for entropy coding [13]. Entropy coding produces the compressed bitstream, which is written to the file following the header information [10].

Conversion from RGB to YCbCr provides the possibility to consider higher compression for color information than luminance information since color information loss has less impact on the image quality. Therefore, YCbCr color space is more suitable for efficient compression. DCT transform and quantization provide a useful statistical structure for compression [14]. The transform separates low and high frequencies, and then, quantization is performed so that low frequencies are represented with more accuracy than higher frequencies because low-frequency variation has much more impact on the visual content of a block [14]. The quantization is the most significant cause of compression, where the compression ratio is specified by the quantization tables for luminance and chrominance. JPEG standard suggests the tables shown in Fig. 2. , but allows the applications to define their own quantization tables. The *independent JPEG group* (IJG) [1] introduces a procedure to determine the tables for a desired

[1] http://www.ijg.org/

978-1-7281-1508-5/19/$31.00 ©2019 IEEE

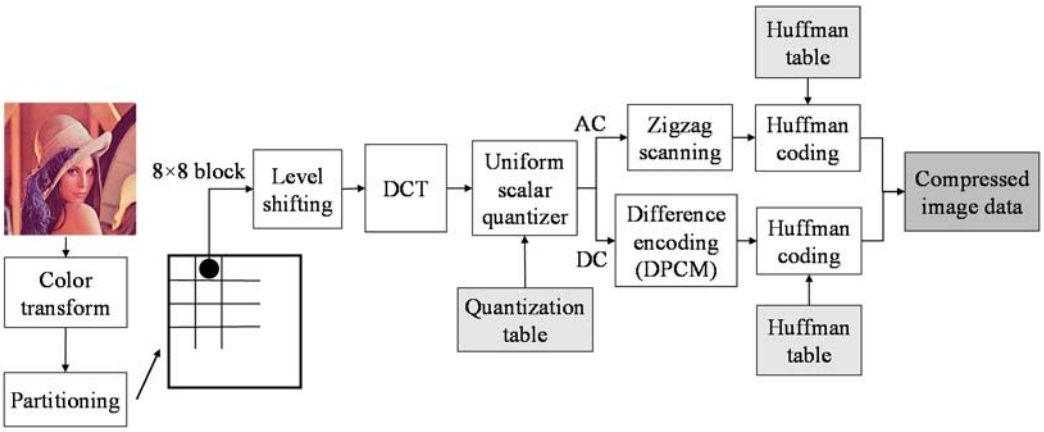

Fig. 1. JPEG encoding procedure

image quality. They define *quality factor* (QF) which is an integer in the range [1, 100], and consider the suggested tables by the standard (T_b) for QF=50. The tables (T_s) for other qualities are computed by scaling T_b as follows [15]:

$$T_s[i,j] = \left\lfloor \frac{S \times T_b[i,j] + 50}{100} \right\rfloor \quad (1)$$

$$where\ S = \begin{cases} \frac{5000}{QF} & QF < 50 \\ 200 - 2QF & QF \geq 50 \end{cases}$$

An example of luminance values of an 8x8 block, level shifting, DCT values, and quantized DCT values for three quality factors is shown in 0The secret data is usually embedded in non-zero quantized DCT coefficients. The higher quantized values (QF=20), the lower image quality, the smaller file size, and the less room for steganography.

The difference between quantized DC coefficient of each block and the DC value of the previous block is Huffman coded. Quantized AC coefficients are encoded differently, where after converting to 1-D array by zigzag scanning of the coefficients in a block, entropy coding starts with zero run length coding and then Huffman coding [13].

Luminance quantization table

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Chrominance quantization table

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	102	63	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Fig. 2. Suggested quantization tables by JPEG standard.

Pixel values

240	211	225	241	189	144	122	116
250	216	226	252	237	157	109	97
247	235	194	220	237	165	128	99
246	247	207	190	160	124	122	75
249	239	230	191	164	136	87	65
246	236	237	199	163	148	88	65
248	228	208	146	150	122	82	62
246	197	134	100	154	121	65	62

Level shifted

112	83	97	113	61	16	-6	-12
122	88	98	124	109	29	-19	-31
119	107	66	92	109	37	0	-29
118	119	79	62	32	-4	-6	-53
121	111	102	63	36	8	-41	-63
118	108	109	71	35	20	-40	-63
120	100	80	18	22	-6	-46	-66
118	69	6	-28	26	-7	-63	-66

DCT values

350.5	435.0	-73.3	58.3	27.0	-12.4	1.4	12.3
129.1	-40.8	-68.3	-34.1	41.9	30.0	-22.2	-9.7
-34.9	-32.1	10.5	17.7	32.9	-7.6	13.3	6.5
24.1	28.6	-18.5	-51.4	-19.6	26.9	29.0	-11.0
-49.8	-1.7	39.7	-2.5	-7.8	4.8	-3.5	-3.8
-5.1	-1.6	27.2	-15.7	-21.1	9.8	-13.3	0.4
7.9	-20.6	-2.0	13.6	4.2	-5.4	-3.3	-8.6
4.1	-4.9	-0.9	7.2	3.7	-11.0	2.3	4.5

QF=20

9	16	-3	1	0	0	0	0
4	-1	-2	-1	1	0	0	0
-1	-1	0	0	0	0	0	0
1	1	0	-1	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

QF=50

22	40	-7	4	1	0	0	0
11	-3	-5	-2	2	1	0	0
-2	-2	1	1	1	0	0	0
2	2	-1	-2	0	0	0	0
-3	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

QF=80

58	109	-18	10	3	-1	0	1
26	-8	-11	-4	4	1	-1	0
-6	-6	2	2	2	0	0	0
4	4	-2	-4	-1	1	1	0
-7	0	3	0	0	0	0	0
-1	0	1	-1	-1	0	0	0
0	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

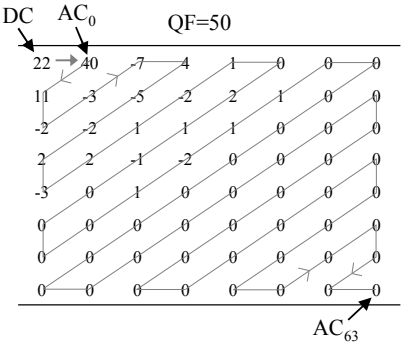

Fig. 3. An 8x8 pixel block, level shifting, DCT transform, and quantized DCT transform for three quality factors 20, 50, and 80.

The zigzag scanning facilitates the entropy coding because low-frequency coefficients with more nonzero values are located before high-frequency coefficients. In run-length coding, the coefficients are coded as (*runs*, *bits*)(*amplitude*). The value *runs* is the number of zero coefficients (*amplitudes*) between the current and the next nonzero amplitude, and *bits* is the number of bits required for representing *amplitude* [14]. This way, runs of zeros are coded efficiently as there are many zero coefficients. When the rest of the coefficients in the array are zero, an end-of-block code (0,0) is sent, see 0[13].

III. Steganography Methods

In this Section, we briefly introduce the selected JPEG steganography methods that we use in the experiments.

Random LSB replacement or *OutGuess* 0.1 [16] replaces the LSB of a quantized AC DCT coefficient, whose value is not 0 or 1, with a bit of the message. To embed a message with *m* bits, *m* coefficients are selected randomly using a pseudo-random number generator. *OutGuess* 0.2 [17] tries to preserve first-order statistics (image histogram) in order to be undetectable by *chi-square attack* or its generalized version. It first embeds the message bits into the DCT coefficients similar to OutGuess 0.1. LSB flipping causes the number of the occurrences of two consequent coefficients such as 2 and 3 (called *pair of values* or POV) in the histogram become close to each other. OutGuess 0.2 modifies some of the unused coefficients in the embedding process to correct the effect of POV. The correction can be made only if enough unused coefficients are available, which leads to a limitation for the embedding capacity.

F5 [18] takes two different actions comparing to OutGuess 0.1 to increase the security: embedding approach and matrix embedding. Instead of replacing the LSB with the message bit, it decrements the absolute value of the DCT coefficient. For example, embedding the message bit 1 in the coefficients 4 and -4 changes the coefficients to 3 and -3, respectively. This prevents the effect of POV, and therefore, the stego image is not detectable by chi-square attack. Matrix embedding reduces the number of changes to the coefficients. The coefficients 1 and -1 are converted to 0 when embedding the bit 0, and 1, respectively. Since zero coefficients are ignored during the message extraction, the same bit is embedded in the next coefficient. Because of this *shrinkage* of ones (-1 and 1), the number of coefficients with the value zero increases. To eliminate the shrinkage problem of F5, *nsF5* (no-shrinkage F5) [19] uses *wet paper codes* instead of matrix embedding. Using wet paper codes, the zeros that are produced by embedding the bit 0 in 1 and -1 can be used for extracting the message, and we do not need to repeat embedding 0 in 1 and -1 coefficients.

In *complementary embedding* (CE) method [20], the DCT coefficients are divided into two parts in order to apply different embedding operations to each part. The message bits are similarly split into two parts. The idea is that one embedding algorithm compensates changes to the image statistics that have been made by the other algorithm. The first algorithm decrements while the second one increments the coefficients by one.

Perturbed quantization (PQ) [21] method uses the values of unquantized DCT coefficients as side information, which is not available to the decoder, in order to decrease the image distortion. The coefficients whose their fractional part is in the range [0.5-ε, 0.5+ ε] are used for embedding message bits. This selection of coefficients provides the opportunity to use wet paper codes in which the embedding channel is not shared with the decoder. Using wet paper codes, the number of changes required for embedding a message is reduced.

J-UNIWARD and *SI-UNIWARD* [4] are two adaptive steganography methods which are designed to minimize an embedding distortion function called *universal wavelet relative distortion* (UNIWARD). The distortion function provides different costs for the coefficients to be selected for embedding the message bits. The complex regions of the image are selected with higher probability than soft regions. Having the cost for each coefficient, *syndrome-trellis codes* (STCs) are used to solve the problem so that the total distortion is minimized. SI-UNIWARD utilizes the side information in the encoder side, where the unquantized DCT coefficients are available when converting a raw image to JPEG. J-UNIWARD and SI-UNIWARD employ all DCT coefficients including DCs and zeros.

IV. Experiments

We experimentally evaluate the impact of 8 well-known JPEG steganography methods on the file size.

We made a database of 100 images which are randomly selected from BOSSbase 1.01 database. The input raw images are given to the JPEG encoder in order to produce JPEG cover images. We use MATLAB programs for generating stego images. All the selected steganography methods get JPEG cover images as the input except SI-UNIWARD and PQ whose input is raw images as they need unquantized DCT coefficients as side information. We produced stego images with eight payloads: 1%, 2%, 5%, 10%, 20%, 30%, 40%, and 50%, and four quality factors: 50, 65, 80, and 95. In total, 24852 stego images were generated. OutGuess 0.2 has a limitation to produce high payloads such as 40% and 50%, and this limitation becomes more as the quality factor increases. That is why the number of stego images is less than expected.

(22, 40, 11, -2, -3, -7, 4, -5, -2, 2, -3, 2, 1, -2, 1, 0, 2, 1, -1, 0, 0, 0, 0, 1, -2, 1, 1, $\underbrace{0, 0, \ldots, 0}_{37 \text{ zeros}}$)

(5)(22)(0, 6)(40), (0, 4)(11), (0,3)(-2),(0,3)(-3),(0,3)(-7), (0,3)(4), (0,3)(-5),(0,3)(-2),(0,3)(2),(0,3)(-3), (0,3)(2), (0,1)(1),(0,3)(-2),(1,1)(1), (0,3)(2),(0,1)(1), (4,2)(-1), (0,1)(1), (0,3)(-2), (0,1)(1), (0,1)1, (0,0)

Fig. 4. Zigzag scanning, and run-length coding of the example 8x8 quantized DCT block in Fig. 3 when QF=50

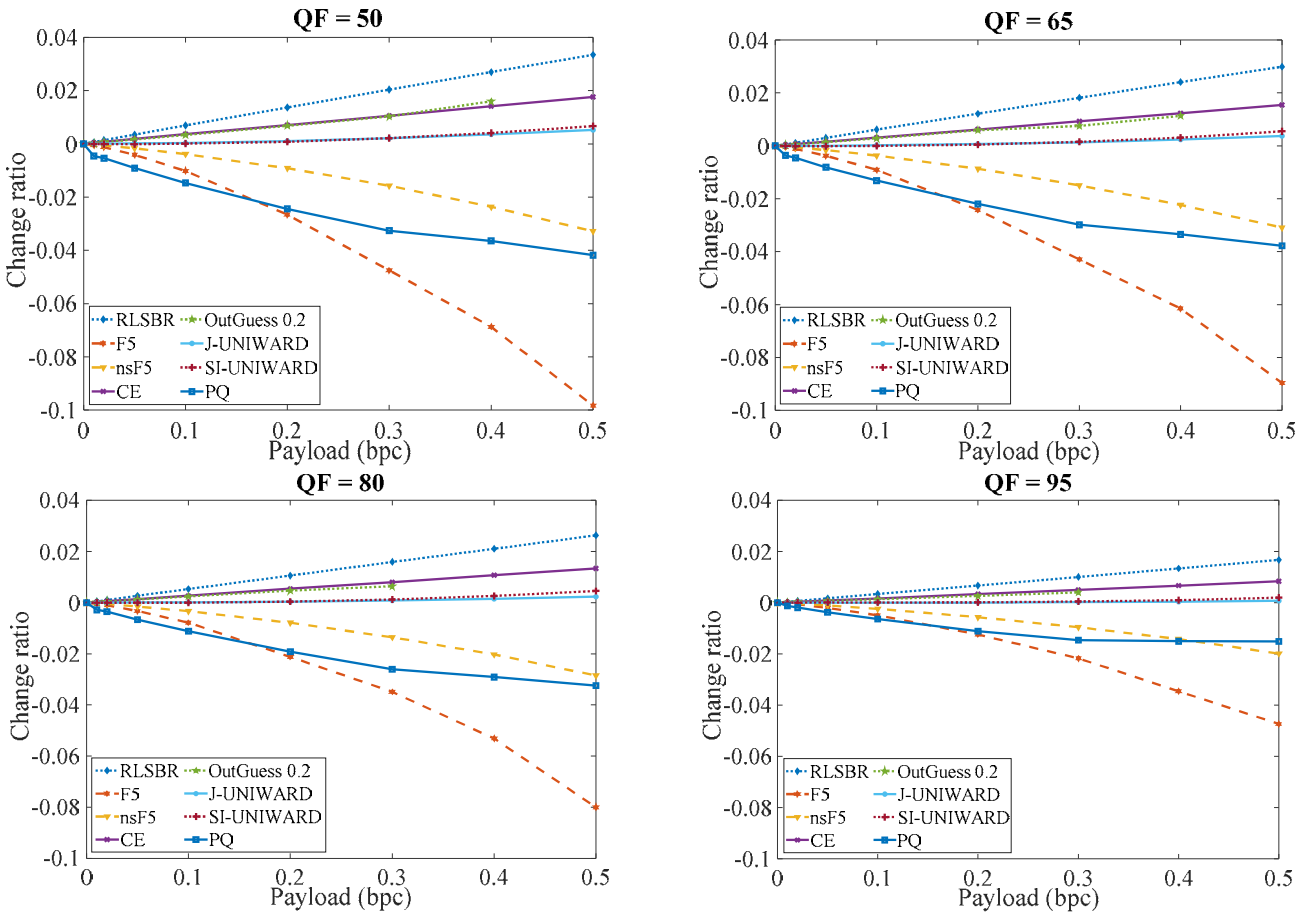

Fig. 5. Impact of the selected steganography methods on JPEG file size in different quality factors as the payload increases

We obtain the size of the compressed bitstream (the file excluding the header part) for a cover and its stego image file, and report the *change ratio*, which is the difference in their size divided by the size of the cover image.

0shows the impact of steganography on the file size for four quality factors. For each payload and each method, the change ratio of 100 images is averaged. The size is affected most by OutGuess 0.1 and F5 methods while J-UNIWARD and SI-UNIWARD result in minimum changes in the file size.

OutGuess 0.1 linearly increases the file size as the payload increases. The change ratio is larger for lower quality factors. To explain the cause of the increase, we give an example on the pair of values 2 and 3, which are converted to 3 and 2 by embedding the message bits 1 and 0, respectively. Since there are always more 2s than 3s among the coefficients, and the coefficients are selected randomly, more 2s are used for embedding, and therefore, after embedding the message bits, the number of 2s and 3s decreases and increases, respectively. Since Huffman coding uses more bits for encoding a larger number, the length of compressed data increases.

As shown in 0, the change ratio for OutGuess 0.2 and CE are almost the same and less than OutGuess 0.1. It is because of the histogram correction made by OutGuess 0.2 and CE. After the correction, the number of values in a pair such as 2 and 3 becomes close to one of the cover image, and therefore, Huffman should be able to compress the stego image as well as cover image. However, the correction is usually not perfect, and there is still an increase in the file size.

F5 linearly reduces the file size as the payload increases, and it has the most impact on the file size comparing to the other steganography methods. Producing many zero coefficients by F5 leads to a decrease in the file size because Huffman coding of JPEG encodes the zeros efficiently. Because of eliminating the shrinkage phenomenon of F5, the decrease in the file size by nsF5 is significantly less than the decrease by F5. The decrease made by nsF5 is because the number of zeros for the stego image is still higher than the number for the cover image. The reason is that both 1 and -1, which exist in the DCT coefficients more than other non-zero coefficients, are converted to 0 when embedding 0 and 1, respectively. Therefore, nsF5 is expected to reduce the file size, but not as much as F5.

PQ is another steganography method that reduces the file size, see Fig. 5. It increases the number of zero coefficients, and that is why the file size decreases. In the embedding process, the coefficients with the value one are converted to zero when embedding the message bit 0. Since there is always a significant number of ones among the DCT coefficients, there would be a significant increase in the number of zeros and consequently a decrease in the file size. Wet paper codes provide the possibility to use also the coefficients 1 for embedding. We remind that OutGuess 0.1 does not use the coefficients 1. Suppose that a one is converted to zero when embedding the message bit 0. Then, the decoder cannot distinguish between this zero coefficient and other zeros which were not used in embedding.

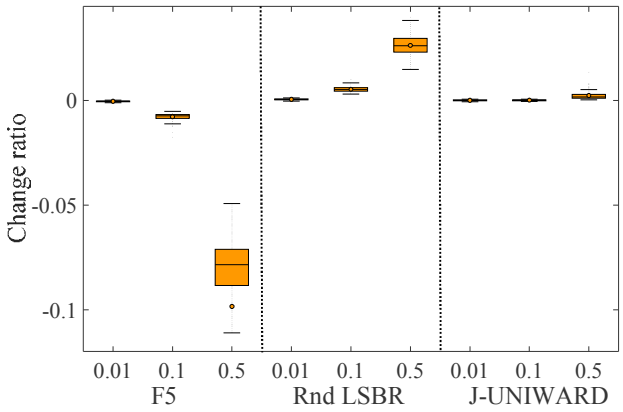

Fig. 6. Variance of change in file size for 100 images

J-UNIWARD and SI-UNIWARD, which are two of the most secure steganography methods so far, have a very little impact on the file size, and almost preserve the file size of the cover image file. We think of two reasons for this. First, they hide the message mostly in complex regions. This means changing the large DCT coefficients, which have a small effect on Huffman coding. Second, STC makes it possible to embed a message with minimum change in the coefficients. J-UNIWARD and SI-UNIWARD similarly as PQ use the coefficients 1 for embedding message bits, but unlike PQ the number of zeros does not increase significantly. The reason is that J-UNIWARD and SI-UNIWARD employ distortion functions in which the costs for converting ones to zeros are high, and it causes less ones are selected for embedding.

0shows the variance of the change ratio among the 100 images of our database. An interesting result about J-UNIWARD (same for SI-UNIWARD) is that the increase in the file size for all images, even in the payload 0.5, is very small. Therefore, the change in file size cannot be used as a feature for steganalysis of J-UNIWARD and SI-UNIWARD while we expect that the file size can be used as a steganalysis feature for other methods especially F5 and OutGuess 0.1.

V. Conclusions

We have conducted an experimental study on the impact of selected steganography methods on JPEG file size. Overall, our experiments show that different steganography methods have different impacts on the file size. F5, nsF5, and PQ reduce the file size while OutGuess 0.1, OutGuess 0.2, complementary embedding increase the file size. J-UNIWARD and SI-UNIWARD preserve the file size of the cover image. In the future, we plan to study how the change in the file size can be used as a feature for detecting steganography content.

References

[1] K. Wang, Z.-M. Lu, and Y.-J. Hu, "A high capacity lossless data hiding scheme for JPEG images," Journal of systems and software, vol. 86, no. 7, pp. 1965-1975, 2013.

[2] F. T. Wedaj, S. Kim, H. J. Kim, and F. Huang, "Improved reversible data hiding in JPEG images based on new coefficient selection strategy," EURASIP Journal on Image and Video Processing, vol. 2017, no. 1, p. 63, 2017.

[3] J. Fridrich, Steganography in digital media: principles, algorithms, and applications. Cambridge University Press, 2009.

[4] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," EURASIP Journal on Information Security, vol. 2014, no. 1, p. 1, 2014.

[5] J. Kodovský and J. Fridrich, "Steganalysis of JPEG images using rich models," in IS&T/SPIE Electronic Imaging,pp. 83030A-83030A-13: International Society for Optics and Photonics, 2012.

[6] V. Holub and J. Fridrich, "Low-complexity features for JPEG steganalysis using undecimated DCT," IEEE Transactions on Information Forensics and Security, vol. 10, no. 2, pp. 219-228, 2015.

[7] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," IEEE Transactions on Information Forensics and Security, vol. 7, no. 3, pp. 868-882, 2012.

[8] F. Huang, X. Qu, H. J. Kim, and J. Huang, "Reversible data hiding in JPEG images," IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, no. 9, pp. 1610-1621, 2016.

[9] B. G. Mobasseri, R. J. Berger, M. P. Marcinak, and Y. J. NaikRaikar, "Data embedding in JPEG bitstream by code mapping," IEEE Transactions on Image Processing, vol. 19, no. 4, pp. 958-966, 2010.

[10] M. Rezaei and M. B. Salahshoor, "Content-independent steganography and steganalysis of JPEG images," in 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2018, pp. 1-6.

[11] G. K. Wallace, "The JPEG still picture compression standard," Communications of the ACM, vol. 34, no. 4, pp. 30-44, 1991.

[12] E. Hamilton, "JPEG File Interchange Format Version 1.02," 1992.

[13] T. Recommendation, "CCITT T. 81," 1993.

[14] G. Hudson, A. Léger, B. Niss, and I. Sebestyen, "JPEG at 25: Still Going Strong," IEEE MultiMedia, vol. 24, no. 2, pp. 96-103, 2017.

[15] J. D. Kornblum, "Using JPEG quantization tables to identify imagery processed by software," digital investigation, vol. 5, pp. S21-S25, 2008.

[16] N. Provos and P. Honeyman, "Hide and seek: An introduction to steganography," IEEE security & privacy, vol. 99, no. 3, pp. 32-44, 2003.

[17] N. Provos, "Defending Against Statistical Steganalysis," in Usenix security symposium vol. 10, pp. 323-336, 2001

[18] A. Westfeld, "F5—A Steganographic Algorithm," in Information Hiding: 4th International Workshop, IH 2001, Pittsburgh, PA, USA, April 25-27, 2001. Proceedings,vol. 2137, p. 289, 2001.

[19] J. Fridrich, T. Pevný, and J. Kodovský, "Statistically undetectable jpeg steganography: dead ends challenges, and opportunities," in Proceedings of the 9th workshop on Multimedia & security, 2007, pp. 3-14: ACM.

[20] C.-L. Liu and S.-R. Liao, "High-performance JPEG steganography using complementary embedding strategy," Pattern Recognition, vol. 41, no. 9, pp. 2945-2955, 2008.

[21] J. Fridrich, M. Goljan, and D. Soukal, "Perturbed quantization steganography with wet paper codes," in Proceedings of the 2004 workshop on Multimedia and security, 2004, pp. 4-15: ACM.