Proceedings of the of the 17th Conference on Computer Science and Intelligence Systems pp. 545–549
DOI: 10.15439/2022F217
ISSN 2300-5963 ACSIS, Vol. 30

# Encrypting JPEG-compressed Images by Substituting Huffman Code Words

Marek Parfieniuk
*University of Bialystok, Institute of Computer Science*
ul. Konstantego Ciolkowskiego 1M, 15-245 Bialystok, Poland
Email: marek.parfieniuk@uwb.edu.pl

***Abstract*—This paper presents a method for encrypting JPEG-coded images that preserves both compression ratio and format of a bit stream. Such solutions allow for selectively hiding information: image contents can be encrypted, while in-file metadata remain readable. Our algorithm is a symmetric, polygram substitution cipher, as it replaces Huffman code words and rearranges value bits that describe the main results of the Discrete Cosine Transform (DCT) of a pixel block: the DC coefficient and the first non-zero AC coefficient. Both length and format of a file are preserved, because bits are modified under constraints on their numbers. Such encryption is a kind of post-processing of a compressed bit stream, and thus it can be built on the top of an existing JPEG codec, without accessing its internals. Compared to previous similar solutions, our approach better hides image contours, exchanging AC for DC energy. Our work also reveals some properties of Huffman code tables and bit streams related to the JPEG standard.**

## I. Introduction

In recent years, one can notice research efforts to combine the JPEG standard for image coding with data encryption [1]. Format-preserving approaches to joint encryption-compression are less efficient, more difficult to implement, and less secure than general-purpose ciphers. Nevertheless, they are useful, allowing for selective encryption, i.e. for protecting only a subset of information contained in a file [2].

In this paper, we propose a method for formant-compliant encryption of JPEG files that is based on substituting and restructuring pairs of variable-length code words under constraints on the total number of bits in a pair. These modifications are made to only code words related to the main results of the Discrete Cosine Transform (DCT) and quantization: to the DC coefficient and to the first non-zero AC coefficient.

As a couple of data units is replaced in accordance with a cryptographic key, our solution is a polygram substitution cypher. Consisting in post-processing of encoding results, it can be built on the top of an existing JPEG codec, without accessing its internals.

Similar known approaches, [3]–[5], modify DC coefficients separately from AC ones. So they have little effects on image contours, unless coefficients are exchanged among blocks of $8 \times 8$ pixels. Our solution is able to exchange DC for AC energy of one image block, so as to better hide the latter, buffering the minimum number of bits.

This paper additionally reveals some new facts on the default Huffman dictionaries and on statistics of code words in JPEG bit streams.

## II. Entropy coding of DCT coefficients in the JPEG standard

The JPEG standard specifies that an image to be compressed is divided into blocks of $8 \times 8$ pixels, and each block is converted into a vector of 64 coefficients, which then are entropy coded. The conversion consists in computing the 2-D DCT of the block, quantizing the resulting matrix, and taking quantized elements in the zig-zag order.

The first of $8 \times 8$ DCT outputs is called the DC (direct-current) coefficient, as it is the average value of all pixels of a block. The remaining 63 outputs are called AC (alternate-current) coefficients, because they reflect deviations from the average value, of various frequencies.

For two consecutive blocks of $8 \times 8$ pixels, the average pixel intensities, or the DC coefficients, often have similar values. Therefore, it is advantageous to encode the difference between them, by using two variable-length code words. The first one has a length of $2 \leq h_0 \leq 9$ bits and results from Huffman encoding of $0 \leq v_0 \leq 11$, the index of the value category that embraces the difference value. The second code word comprises $v_0$ bits which point out a particular value in the category. The first bit represents the sign of the difference, whereas the remaining bits determine its magnitude.

Table I lists the categories and Huffman code words used to encode DC coefficients. The lower magnitudes of the values that comprise a category, the fewer values in this category. So, a shorter the code word is assigned to its index, in order to achieve data compression.

A non-zero quantized AC coefficient is usually preceded by a series of zero-valued ones. Thus its value is encoded together with the number of the latter, by using two code words. For the $k$th non-zero coefficient, in the zig-zag order, the first word comprises $2 \leq h_k \leq 16$ bits and is obtained by Huffman encoding of the $(r_k, v_k)$ pair, where $0 \leq r_k \leq 15$ is the number of the zero-valued AC coefficients that precede this non-zero one, and $1 \leq v_k \leq 10$ is the index of the category that embraces the value of this coefficient. The second word comprises $v_k$ bits, which determine a value in the $v_k$th category. As zero-valued quantized AC coefficients often occur in long runs, and non-zero ones often have small magnitudes, compression can be achieved by assigning shorter words to $(r_k, v_k)$ pairs that describe such occurrences.

Two special Huffman code words occur without value bits.

TABLE I
VALUE CATEGORIES AND HUFFMAN CODE WORDS FOR ENCODING DIFFERENCES BETWEEN DC COEFFICIENTS OF IMAGE LUMINANCE

| $v_0$ | Value range | Huffman code word | $h_0$ |
|---|---|---|---|
| 0 | 0 | 00 | 2 |
| 1 | $\pm 1$ | 010 | 3 |
| 2 | -3,-2, 2, 3 | 011 | 3 |
| 3 | $\pm(4, \ldots, 7)$ | 100 | 3 |
| 4 | $\pm(8, \ldots, 15)$ | 101 | 3 |
| 5 | $\pm(16, \ldots, 31)$ | 110 | 3 |
| 6 | $\pm(32, \ldots, 63)$ | 1110 | 4 |
| 7 | $\pm(64, \ldots, 127)$ | 11110 | 5 |
| ... | ... | ... | ... |
| 11 | $\pm(1024, \ldots, 2047)$ | 111111110 | 9 |

The EOB (End-of-Block) word is placed after the codes of the last non-zero AC coefficient, so as to tell that the remaining ones are zero. The ZRL (Zero-run-length) word represents a series of 16 zero AC coefficients between non-zero ones.

## III. ENCRYPTING JPEG BIT STREAMS BY SUBSTITUTING HUFFMAN CODE WORDS

### A. Substitution idea and constraints

Polygram substitution ciphers replace several symbols of a text with the same number of other letters. Following this idea, we have shown in [6] that Huffman-encoded data can be encrypted by substituting pairs of code words. Herein, we adapt this approach to JPEG bit streams, in which Huffman code words are interleaved with value bits, and thus they need to be substituted somewhat tricky. Moreover, it is pointless to modify all code words, as the contents of an image is described primarily by the main quantized coefficients of the DCT of an $8 \times 8$ pixel block: the DC one and the first non-zero AC one.

So, we propose to encrypt JPEG-encoded images by processing them block-by-block, by substituting Huffman code words that describe categories of the aforementioned coefficients and by moving bits between the code words that describe coefficient values. As the cipher should not enlarge files, we allow only such substitutions that

$$\underline{h}_0 + \underline{h}_1 = h_0 + h_1 \quad \text{and} \quad \underline{v}_0 + \underline{v}_1 = v_0 + v_1 \qquad (1)$$

where underlines denote the lengths of code words that replace the original ones. These constraints ensure that a substitution changes neither the total length of Huffman code words nor the total number of value bits, $\underline{h}_0+\underline{h}_1+\underline{v}_0+\underline{v}_1 = v_0+v_1+h_0+h_1$.

The constraints can be explained by using Table II. It lists 12-bit, equal-length combinations of DC- and AC-related code words and value bits. The combinations have been grouped with respect to the total number of the value bits, $v_0 + v_1$. A combination can be substituted for each other of the same group, but not for one of another group.

### B. Encryption procedure

Both encryption and decryption of our cipher can be explained by using the data flow shown in Fig. 1.

Assuming that the method is applied to an existing JPEG bit stream, the first step is to scan the stream in order to determine the Huffman code words and value bits that describe the DC and AC coefficients of interest of a subsequent block of $8 \times 8$ pixels. If the method has to be integrated with a JPEG encoder, than it can be applied to quantized coefficients before forming a bit stream. Knowing the code words, or coefficient values, one can determine $(h_1 + h_0)$ and $(v_1 + v_0)$ and decide whether substitution is possible. If so, then these bit counts point out the group of $N >= 2$ pairs of code words that can be exchanged for each other in accordance with (1).

TABLE II
ALL 12-BIT COMBINATIONS OF DC- AND (AFTER "-") AC-RELATED HUFFMAN CODE WORDS AND VALUE BITS (DENOTED AS "X", BEING 0 OR 1), GROUPED WITH RESPECT TO THE TOTAL NUMBER OF VALUE BITS

| DC-AC code words | $h_0$ | $h_0$ | $v_0$ | $v_1$ | $r_1$ | $v_0 + v_1$ | $h_0 + h_1$ |
|---|---|---|---|---|---|---|---|
| 00-111111000x | 2 | 9 | 0 | 1 | 8 | | |
| 00-111111001x | 2 | 9 | 0 | 1 | 9 | 1 | 11 |
| 00-111111010x | 2 | 9 | 0 | 1 | 10 | | |
| 00-11111001xx | 2 | 8 | 0 | 2 | 2 | | |
| 010x-1111010x | 3 | 7 | 1 | 1 | 5 | 2 | 10 |
| 010x-1111011x | 3 | 7 | 1 | 1 | 6 | | |
| 00-1111001xxx | 2 | 7 | 0 | 3 | 1 | | |
| 011xx-111010x | 3 | 6 | 2 | 1 | 3 | 3 | 9 |
| 011xx-111011x | 3 | 6 | 2 | 1 | 4 | | |
| 011xx-11011xx | 3 | 5 | 2 | 2 | 1 | 4 | 8 |
| 100xxx-11100x | 3 | 5 | 3 | 1 | 2 | | |
| 00-11010xxxxx | 2 | 5 | 0 | 5 | 0 | | |
| 011xx-1011xxx | 3 | 4 | 2 | 3 | 0 | 5 | 7 |
| 101xxxx-1100x | 3 | 4 | 4 | 1 | 1 | | |

Let us assume that the members of the substitute group are ordered, so that they form an array and can be indexed by the numbers from 0 to $(N - 1)$. If $i_{\text{original}}$ is the index of the pair of original code words, than by adding (modulo $N$) a pseudorandom offset we obtain the index of a substitute pair:

$$i_{\text{substitute}} = \text{mod}(i_{\text{original}} + \text{randi}(N), N) \qquad (2)$$

The "mod" function gives the reminder after division, and is used to ensure that $0 \leq i_{\text{substiture}} < N$, like $i_{\text{original}}$. The "randi" produces a pseudorandom integer from the discrete uniform distribution over $[1, N]$. This function is assumed to use a pseudorandom number generator whose output can be controlled by using the cryptographic key as the seed.

The Huffman code words determined by $i_{\text{substitute}}$ are submitted to the output (encrypted) bit stream, followed by the original value bits, rearranged in accordance with $\underline{v}_0$ and $\underline{v}_1$ that match the substitutes. The cipher does not affect bits that describe the rest of AC coefficients of the given pixel block.

### C. Decryption procedure

By detecting and analysing a pair of DC- and AC-related code words of the encrypted bit stream, one can determine the substitute group and $i_{\text{substitute}}$. The original code words are pointed by the inverse of (2)

$$i_{\text{original}} = \text{mod}(i_{\text{substitute}} - \text{randi}(N), N) \qquad (3)$$

provided that, at both encryption and decryption sides of a flow of JPEG-compressed pictures, (i) members of a substitute group are ordered in the same way, (ii) the same cryptographic key is used as the seed of the pseudorandom generator behind the "randi" function, when the processing of a bit stream starts, and (iii) $8 \times 8$ pixel blocks are processed in the same order.

Obviously, in addition to recovering the Huffman code words, it is necessary to accordingly rearrange the value bits.

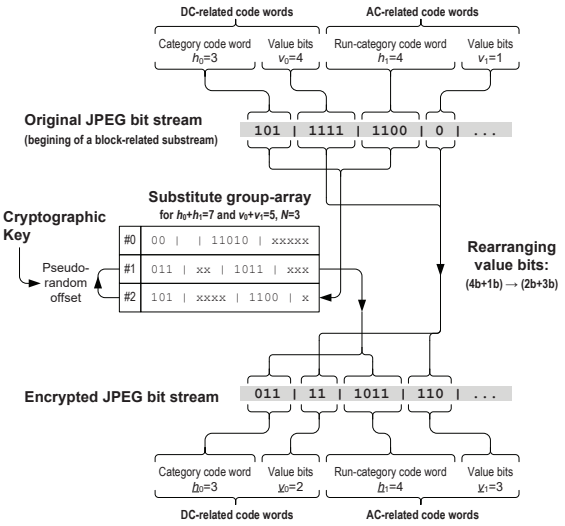


Fig. 1. Encryption of JPEG bit streams by length-constrained substitution of Huffman code words and rearrangement of value bits.

## IV. Substitution Limitations

For most images and quality settings, code words can be substituted sufficiently often to make it impossible to recover a readable, quality image from an encrypted bit stream, without knowing the cryptographic key. However, when an image in encoded with low-quality settings, pixel blocks might occur to which the proposed cipher cannot be applied. The default Huffman dictionaries determine not so many pairs of code words that have no substitutes, but the majority of the replaceable combinations of bits occur only occasionally when compressing natural images.

### A. Substitution limitations by Huffman code tables

The JPEG standard specifies the default dictionaries of Huffman code words. The tables for luminance contain 12 words for encoding differences between DC coefficients and 161 words for encoding AC coefficients. These words can be combined into $12 \times 161 = 1932$ DC-AC pairs, which can be grouped with respect to both the total length of Huffman code words and the total number of the value bits that must accompany them. If a resulting group comprises two or more pairs of code words, then these DC-AC pairs can be substituted for each other in accordance with (1).

The sizes of the groups can be visualized by colors as in Fig. 2. The bit numbers related to the axes determine a group, and are coordinates of the small square whose color reflects the number of pairs in this group, in accordance with the legend.

Groups exist such that a pair of code words has as many as several dozen of substitutes. But some groups comprise only one combination of Huffman code words, which cannot be substituted. The great majority of the pairs, 1892 (98%) of them, have at least one potential substitute.

Table III lists the DC-AC pairs of code words that cannot be substituted. Moreover, the cipher cannot be applied to blocks for which all quantized AC coefficients are zero, i.e. when the DC-related bits are followed by the EOB special code word.

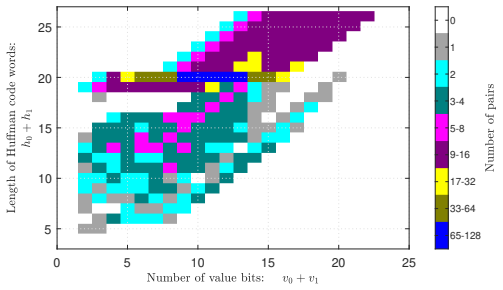


Fig. 2. Numbers of DC-AC pairs of default code words that can be substituted in accordance with (1).

TABLE III
Pairs of DC- and (after "-") AC-related default code words that cannot be substituted in accordance with (1)

| # | DC- and AC-related code words and value bits (denoted as "x") | $h_0$ | $v_0$ | $h_1$ | $v_1$ | $r_1$ | $h_0$ + $h_1$ | $v_0$ + $v_1$ | $h_0 + v_0$ + $h_1 + v_1$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 00-00x | 2 | 0 | 2 | 1 | 0 | 4 | 1 | 5 |
| 2 | 00-01xx | 2 | 0 | 2 | 2 | 0 | 4 | 2 | 6 |
| 3 | 010x-00x | 3 | 1 | 1 | 2 | 0 | 5 | 2 | 7 |
| 4 | 011xx-1100x | 3 | 2 | 4 | 1 | 1 | 7 | 3 | 10 |
| 5 | 100xxx-1100x | 3 | 3 | 4 | 1 | 1 | 7 | 4 | 11 |
| 6 | 011xx-100xxx | 3 | 2 | 3 | 3 | 0 | 6 | 5 | 11 |
| 7 | 100xxx-100xxx | 3 | 3 | 3 | 3 | 0 | 6 | 6 | 12 |
| 8 | 110xxxxx-01xx | 3 | 5 | 2 | 2 | 0 | 5 | 7 | 12 |
| 9 | 100xxx-1011xxxx | 3 | 3 | 4 | 4 | 0 | 7 | 7 | 14 |
| 10 | 100xxx-11010xxxxx | 3 | 3 | 5 | 5 | 0 | 8 | 8 | 16 |
| 11 | 1111110xxxxxxxxx-00x | 7 | 9 | 2 | 1 | 0 | 9 | 10 | 19 |
| 12 | 101xxxx-1111000xxxxxx | 3 | 4 | 7 | 6 | 0 | 10 | 10 | 20 |

### B. Substitution limitations by image contents

The essence of the JPEG standard lays in the adjustment of entropy codes to statistics of quantized DCT coefficients of an average, natural image. By lowering the quality settings, one quantizes DCT coefficients more roughly, and thus decreases differences between DC coefficients, increases the number and seriality of zero-valued AC coefficients, and decreases magnitudes of non-zero ones. The reconstructed image looks worse, but it is described by a shorter bit stream.

Shortening of code words results in an increased probability that they cannot be replaced in accordance with our cipher. Moreover, it decreases the number of substitutes, when code words can be replaced. Both these issues can be explained by using Table IV and Fig. 3, which show properties of bit streams that result from JPEG-encoding of the Lena image for various quality settings.

In Table IV, one can see that decreasing quality increases the number of pixel blocks which are described by only DC-related bits, followed by the EOB code word, so that our cipher cannot be applied. Such situations do not occur for higher-quality settings, $Q > 75\%$, but become frequent for $Q < 50\%$.

In Fig. 3, the color of a small square shows how many blocks of an image are described by code words and value bits, whose lengths and numbers, respectively, are given by the coordinates of the square. When the quality is decreased, the distribution of code words shifts toward the origin: shorter ones occur more frequently, while longer ones disappear.

The problem becomes clear after comparing the plots in Fig. 3 to that in Fig. 2. The former overlap little with the

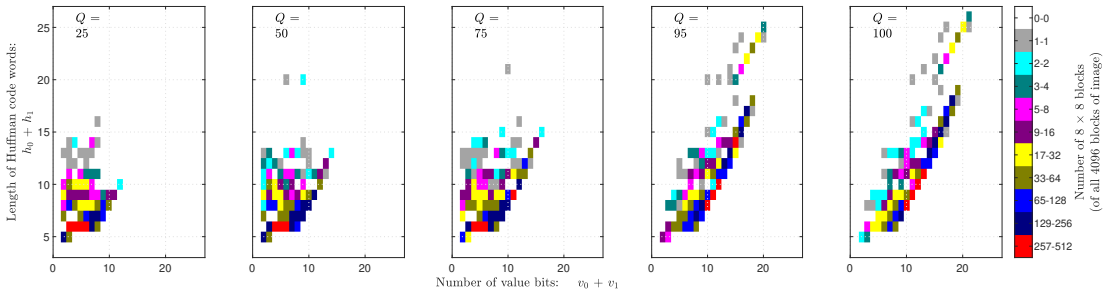


Fig. 3. Occurrence frequencies of DC-AC pairs of code words for the Lena image ($512 \times 512$ pixels).

TABLE IV
PROPERTIES OF JPEG BIT STREAMS OF THE LENA IMAGE

| Quality [%] | Ratio [bpp] | PSNR [dB] | Percentage of blocks with the DC coefficient followed by $N$ nonzero AC coefficients [%] | | | | |
|---|---|---|---|---|---|---|---|
| | | | $N = 0$ | $N = 1$ | $N = 2$ | $N = 3$ | $N \geq 4$ |
| 100 | 4.93 | 47.34 | 0 | 0 | 0 | 0 | 100 |
| 90 | 1.78 | 39.92 | 0 | 0 | 0 | 0.02 | 99.98 |
| 75 | 0.98 | 37.40 | 0.29 | 1.46 | 4.27 | 6.98 | 86.98 |
| 50 | 0.63 | 35.55 | 6.22 | 11.49 | 13.62 | 10.88 | 57.76 |
| 25 | 0.43 | 33.57 | 22.31 | 19.11 | 13.28 | 8.10 | 37.18 |

TABLE V
PERCENTAGES OF $8 \times 8$ PIXEL BLOCKS FOR WHICH THE DC-AC CODE WORDS CAN BE SUBSTITUTED IN ACCORDANCE WITH OUR CIPHER

| | Lena | Baboon | Barbara | Boat |
|---|---|---|---|---|
| Quality [%] | Percentage of blocks [%] | | | |
| 100 | 84.49 | 88.84 | 85.81 | 84.17 |
| 90 | 84.47 | 88.33 | 85.37 | 78.83 |
| 75 | 81.98 | 80.85 | 80.46 | 71.02 |
| 50 | 69.60 | 73.04 | 69.82 | 55.93 |
| 25 | 51.63 | 68.60 | 57.39 | 43.77 |

latter even for higher-quality compression. For lower-quality settings, the overlap is even smaller, and, what is even worse, it occurs at the region, in Fig. 2, that is related to code words that cannot be substituted or belong to substitute groups that comprise very few pairs of code words.

So, in order to evaluate the probability that our cipher can be applied to a pixel block of a given image, for given quality settings, one should determine the corresponding data distributions like those illustrated in Fig. 3 and combine it with that that is shown in Fig. 2.

Table V shows summaries of such evaluations for several well-known test images. In particular, it lists the percentages of $8 \times 8$ blocks to which our cipher can be applied. They are satisfactory: for reasonable quality settings, our cipher is able to modify considerable areas of an image.

A related measure of security is the average number of substitutes per block. For these images, it varies from 1.3–1.9 to 2.9–3.0, for quality settings from 25% to 100%, respectively. By raising these numbers to the power of the number of blocks in an image, one can estimate the number of substitution combinations that must be reviewed in a brute force attack on our cipher. Obviously, it is virtually impossible to recover a quality image, without knowing the secret key.

## V. CIPHER EVALUATION

### A. Conceptual contribution

Our idea is related to encrypting data by altering Huffman tables in accordance with a cryptographic key [7], [8]. However, we noticed no similar solutions, which would be based on modifying both Huffman code words and value bits, of DC and AC DCT coefficients. In most works, AC- and DC-related code words are transformed independently, so encryption-related changes to the bit stream are less deep than in our cipher, unless coefficients are exchanged among blocks of $8 \times 8$ pixels, which requires complicated data buffering.

By modifying together DC and AC coefficients of one pixel block, our method is able to more affect the latter, or image contours, without referring to other blocks.

A value of our works is also in providing arguments to question the claims of [9], in which Huffman coding has been evaluated as being unsuitable to encryption aimed at format compliance and file size preservation.

### B. Cipher manifestation in images and its strength

Figure 4 shows the Lena test image, and results of straightforward (without decryption) decoding of a corresponding JPEG bit stream that had been encrypted using our method. The reconstructed image is unreadable, mainly because of rapid changes in average intensity of pixel blocks.

Unfortunately, an attacker can easily retrieve contours of images from encrypted files. It is sufficient to only set all DC coefficients to zero, and then to decode a picture without care about decryption. Fig. 5 shows results of such decoding of the Lena image from original and encrypted bit streams. Even though edges are reconstructed incorrectly, they appear in correct blocks, forming contours.

This weakness is not specific for our cipher. It characterizes virtually all JPEG-compliant and file-size-preserving approaches to joint compression-encryption, being related to the JPEG coding principle of processing images block-by-block. A cipher cannot modify the contour location without moving information from block to block. So most publications about extending JPEG coding with encryption describe some method of exchanging pixels or (encoded) DCT coefficients among blocks [3], [5]. They destroy contours but at the costs of buffering an entire image and of increasing file size. Our solution could as well be combined with such a method.

Fig. 4. Lena image and corresponding picture decoded without taking into consideration that JPEG bit streams have been encrypted.

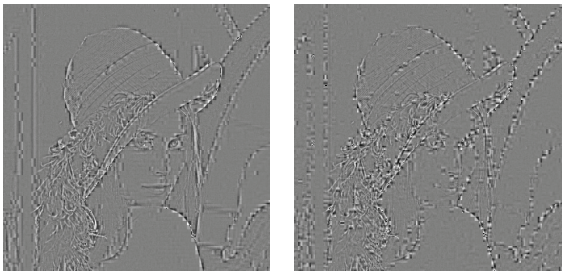

Fig. 5. Non-DC contents of the Lena images: original and decoded without taking into consideration that JPEG bit streams have been encrypted.

### C. Inconsistencies in DC and AC coefficients

Our cipher might produce files that are logically inconsistent. In Fig. 4, artifacts result mainly from that encrypted code words might describe differences that sum up to values of the DC coefficient that are outside the allowed range $\pm 2^{11}$, for luminance. A decoder silently converts an incorrect value into a number in the range, by overflow, or saturation.

The issue is known, and some methods have been proposed to handle it [3], [10], but they are based on simultaneously processing many blocks of $8 \times 8$ pixels. The problem seems to be neglected in some works, in which DC-related Huffman code words are left untouched, and only value bits are modified simply, by pseudorandomly changing only value sign or by XOR-ing more bits with pseudorandom patterns [11], [12].

In most of works that modify the DC coefficient, the file size is not preserved [13]–[15].

A related problem is that encrypted AC code words might describe so long runs of zero-valued AC coefficients, that the 64-element vector is too short to put all coefficients into it. We noticed no publications which would discuss this, even though the issue can be caused by some known ciphers, like those of [4] and [16] that shuffle AC-related code words inside a block and among blocks, respectively.

### D. Computational load and memory consumption

Considerable computations or memory are necessary to determine substitutes of code words. A slower but memory-efficient approach is to determine a substitute on-the-fly, by scanning Huffman tables provided by a JPEG codec. Alternatively, an array of arrays can be prepared that stores information about substitute groups. It would occupy an additional memory much larger than that of the Huffman dictionary, but substitutions could be realized quickly by using $(h_1 + h_0)$ or $(v_1 + v_0)$ as an index into the array of code-word groups.

## VI. Conclusion

From the point of view of practice, our cipher rather only slightly outperforms the known approaches to combining JPEG compression with encryption. However, it can be evaluated as conceptually subtle and sophisticated, being based on nuances related to JPEG Huffman dictionaries of code words and to distributions of quantized DCT coefficients. To the best of our knowledge, this is the first paper that demonstrates and analyses these nuances, via unique plots and tables.

## References


[1] P. Li and K.-T. Lo, "Survey on JPEG compatible joint image compression and encryption algorithms," *IET Signal Process.*, vol. 14, no. 8, pp. 475–488, 2020. doi: 10.1049/iet-spr.2019.0276

[2] A. Massoudi, F. Lefebvre, C. De Vleeschouwer, B. Macq, and J.-J. Quisquater, "Overview on selective encryption of image and video: Challenges and perspectives," *EURASIP J. Inf. Secur.*, vol. 2008, pp. 5:1–5:18, Jan. 2008. doi: 10.1155/2008/179290

[3] J. He, S. Huang, S. Tang, and J. Huang, "JPEG image encryption with improved format compatibility and file size preservation," *IEEE Trans. Multimedia*, vol. 20, no. 10, pp. 2645–2658, 2018. doi: 10.1109/TMM.2018.2817065

[4] S. Auer, A. Bliem, D. Engel, A. Uhl, and A. Unterweger, "Bitstream-based JPEG encryption in real-time," *Int. J. Digital Crime Forensics*, vol. 5, no. 3, pp. 1–14, jul 2013. doi: 10.4018/jdcf.2013070101

[5] K. Kurihara, M. Kikuchi, S. Imaizumi, S. Shiota†, and H. Kiya, "An encryption-then-compression system for JPEG/Motion JPEG standard," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E98.A, no. 11, pp. 2238–2245, nov 2015. doi: 10.1587/transfun.E98.A.2238

[6] M. Parfieniuk and P. Jankowski, "Encrypting Huffman-encoded data by substituting pairs of code words without changing the bit count of a pair," in *Proc. 3rd Int. Conf. Cryptography Security Syst. (CSS)*, Lublin, Poland, 22-24 Sep. 2014. doi: 10.1007/978-3-662-44893-9_2 pp. 12–22.

[7] M. S. Kankanhalli and T. T. Guan, "Compressed-domain scrambler/descrambler for digital video," *IEEE Trans. Consum. Electron.*, vol. 48, no. 2, pp. 356–365, May 2002. doi: 10.1109/TCE.2002.1010142

[8] C.-P. Wu and C.-C. J. Kuo, "Fast encryption methods for audiovisual data confidentiality," in *Multimedia Syst. Appl. III*, ser. Proc. SPIE, vol. 4209, Boston, MA, Nov. 2000. doi: 10.1117/12.420829 pp. 284–295.

[9] S. Li, *On the Performance of Secret Entropy Coding: A Perspective Beyond Security*. Berlin, Heidelberg: Springer, 2012, pp. 389–401.

[10] W. Li and Y. Yuan, "A leak and its remedy in JPEG image encryption," *Int. J. Computer Mathematics*, vol. 84, no. 9, pp. 1367 – 1378, Sep. 2007. doi: 10.1080/00207160701294376

[11] K. Yi and K. Kim, "Encryption method of compressed images with JPEG compliance by shuffling information both in spatial and frequency domains," in *Advanced Multimedia and Ubiquitous Engineering*, J. J. Park, H. Jin, Y.-S. Jeong, and M. K. Khan, Eds. Singapore: Springer, 2016. doi: 10.1007/978-981-10-1536-6_86 pp. 661–667.

[12] S. Li and Y. Zhang, "Quantized DCT coefficient category address encryption for JPEG image," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 4, pp. 1790–1806, Apr. 2016. doi: 10.3837/tiis.2016.04.018

[13] Y. Mao and M. Wu, "A joint signal processing and cryptographic approach to multimedia encryption," *IEEE Trans. Image Process.*, vol. 15, no. 7, pp. 2061–2075, Jul. 2006. doi: 10.1109/TIP.2006.873426

[14] S. Lian, J. Sun, and Z. Wang, "A novel image encryption scheme based-on JPEG encoding," in *Proc. 8th Int. Conf. Inf. Vis. (IV)*, London, UK, 16 Jul. 2004. doi: 10.1109/IV.2004.1320147 pp. 217–220.

[15] X. Niu, C. Zhou, J. Ding, and B. Yang, "JPEG encryption with file size preservation," in *Proc. Int. Conf. Intell. Inf. Hiding Multimedia Signal Process. (IIHMSP)*, Harbin, China, 15-17 Aug. 2008. doi: 10.1109/IIH-MSP.2008.207 pp. 308–311.

[16] B. Yang, C.-Q. Zhou, C. Busch, and X.-M. Niu, "Transparent and perceptually enhanced JPEG image encryption," in *Proc. 16th Int. Conf. Digital Signal Process.*, Santorini, Greece, 5-7 Jul. 2009. doi: 10.1109/ICDSP.2009.5201075 pp. 1–6.