https://doi.org/10.2352/ISSN.2470-1173.2017.7.MWSF-323

How to recompress a JPEG crypto-compressed image?

Vincent Itier, William Puech; LIRMM, UMR 5506 CNRS, University of Montpellier; Montpellier, FRANCE

Abstract

This paper presents a new method, of recompressing a JPEG crypto-compressed image. In this project, we propose a crypto-compression method which allows recompression without any information about the encryption key. The recompression can be executed on the JPEG bitstream directly by removing the last bit of the code representation of a non null DCT coefficient and adapting its Huffman code part. To be able to apply this approach, the quantization table is slightly modified to make up the modifications. This method is efficient to recompress a JPEG crypto-compressed image in terms of ratio compression. Moreover, since the encryption is fully reversible, the decryption of the recompressed image produces an image that has a similar visual quality compared to the original compressed image.

Introduction

Today, there is an enormous amount of images exchanged via the Internet and this number increases every day. For acquisition, exchange, transmission and storage, compressed images are often used. The most popular image compression standard is JPEG [17]. The JPEG File Interchange Format [3] (JFIF) is often used for encapsulate image compressed using JPEG compression. Moreover, the JPEG compression has been more standardised by the Independent JPEG Group [20] (IJG). This method of lossy compression is popular both in normal use and in research. Moreover, for security purpose, it is common to encrypt data during their transit and their storage. Let consider two users Alice and Bob. Alice wants to send a raw image to Bob through a network that she does not trust. Alice can send a encrypted image in order to mask the visual information but bandwidths can be limited. Thus, sending a raw image has a huge cost. A realistic scenario is to consider a compression before sending an image. In order to send a crypto-compressed image, there exists two main scenarios. The first case consists to encrypt the image and to compress it, as done in [16, 6, 21]. In this case the compression is limited due to the removing of redundancy by the encryption. Moreover, the compression has often a specific format. The second case consists to compress the image in a common format and to encrypt it. In order to reinforce the protection and the confidentiality of JPEG compressed images, several JPEG crypto-compression techniques have been proposed. These techniques aim to offer a format compliant visual masking. In order to partially hide details of an image, such as face or text, without losing the image meaning, selective or partial encryption methods have been proposed [14, 18, 12, 5]. In the case where confidentiallity is needed, full encryptions methods [7, 9, 8, 10] allow to efficiently hide the content. We analyzed which encryption method to use in order to have interesting visual masking and a good compression rate. Most of JPEG encryption methods encrypt the AC coefficients of DCT transformation and do not cipher the DC coefficients or with a specific method, as they are encoded differently and contain important visible information [19]. The less sensitive AC coefficients can be encrypted using AES xor encryption scheme [14, 11] or by scrambling technique [7, 5]. However, the network capacity can be more limited at some nodes, that can involves problems during image transmission. In this context, it is common to apply a JPEG recompression onto clear image in order to reduce image size. This recompression may produce block artifacts and has been studied by Bauschke *et al.* [1]. The main problem is to recompress a crypto-compressed image without being exposed to privacy leaks. The best solution is to avoid the decryption and to work in the encrypted domain. Working in encrypted domain is a main challenge concerning privacy and security in many applications such as indexing, retrieving and data hiding. It requires to find an encrypted domain which allows to perform a recompression. Recently, many applications in encrypted domain have been proposed such as features extraction proposed by Hsu *et al.* [4] and watermarking proposed by Qian *et al.* [13].

In this paper, we propose a new method for JPEG recompression in the encrypted domain which preserve the image content. We define a JPEG crypto-compression method which allows the recompression directly in the bitstream. This method is based on xor operation on the appended bits in entropy encoding with a pseudo-random bits sequence as done by Rodrigues *et al.* [14]. The main idea is to divide non null DCT coefficients by removing one bit of the encrypted value and adapting quantization tables. Our approach allows us to localize which bits of the pseudo-random bits sequence are lost after the recompression. Thus, the decryption is fully reversible after the recompression process. The rest of the paper is organized as follow. First, we present the JPEG standard and we discuss previous work in crypto-compression. Then, we describe the proposed JPEG crypto-compression method, and the recompression method. The experimental results are presented and analyzed. Finally, we provide a conclusion of this study.

Related Work

In this Section, we first describe JPEG compression and more precisely JPEG quantization and JPEG entropy coding. Then, we present previous JPEG encryption methods and we explain their limits in terms of confidentiality, security and bitstream size.

JPEG compression

According to JPEG standard, a RGB image represented by three components red, green, blue, is first converted into luminance/chrominance space (YUV). Fig. 1 illustrates the main stages of JPEG compression for the luminance component. For the rest of this study, we consider only the luminance component. First, an image I is decomposed into non-overlapped 8×8 pixel blocks. Each block is transformed into DCT coefficients. A quantization matrix is defined depending on the desired compres-

sion. Then the quantization matrix is used to quantify DCT coefficients. Then each block defines a Minimum Code Unit (MCU) which is compressed using entropy encoding. Finally, the JPEG compressed image I_{QF} consists of a header followed by the MCUs codes.

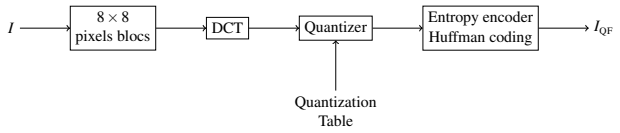


Figure 1: JPEG Encoder.

The decoding process of a JPEG decoder starts by the extraction of the MCUs from entropy decoding. Coefficients are dequantified with the same table and an Inverse DCT (IDCT) is applied to get blocks of 8×8 pixel values. The proposed method is based onto two main steps JPEG quantization, and JPEG entropy encoding, which are described in details in the next sections.

JPEG quantization

The DCT transformation is not involved in the compression. At the contrary, it produces 64 non correlated floating coefficients for each 8×8 pixel block. The compression is mostly performed by the entropy encoding. Nevertheless, the quantization aims to code DCT coefficients onto small integers and to produce zero coefficients which are efficiently encoded. The quantization step is the cause of losses in JPEG compression, the other stages are reversible. In the most commun case of JFIF format, quantization matrices can be transmitted in JPEG header since, generally, each encoder has its own specification. Generally, quantization tables quantify less low frequency coefficients which are more significant for the visual human system. However, in the standard recommanded by the IJG, the quantization strenght is given by a quality factor $QF \in [1,100]$. This quality factor is used to compute the quantization table T_{QF}. The IJG specifies a standard quantization table T_{50} for a quality factor QF of 50%. From this table, each quantization table with a quality factor $QF \in [1,100]$ can be calculated. First, the corresponding the scaling factor SF is given as:

$$SF = \begin{cases} 5000/\mathrm{QF}, & \text{if } \mathrm{QF} < 50 \\ 200 - \mathrm{QF}, & \text{otherwise.} \end{cases} \tag{1}$$

Then, the new quantization table T_{QF} is computed for all index pair $(u,v),\ u,v \in \{0,7\}$:

$$q^{\mathrm{QF}}_{(u,v)} = \left\lfloor \frac{q^{50}_{(u,v)} \cdot SF + 50}{100} \right\rfloor. \tag{2}$$

Furthermore, in order to fulfill the IJG recommendation and for full JPEG baseline compatibility, the quantization matrix values are constrained to the range $[1,255]$:

$$q^{\mathrm{QF}}_{(u,v)} = \begin{cases} 1 & \text{if } q^{\mathrm{QF}}_{(u,v)} < 1 \\ 255 & \text{if } q^{\mathrm{QF}}_{(u,v)} > 255,\ 1u,v \in \{0,7\}. \\ q^{\mathrm{QF}}_{(u,v)} & \text{otherwise.} \end{cases} \tag{3}$$

This may have an effect for quality factors below 25, and it prevents very low quality files from being generated [20]. Finally, block quantization is defined as the integer division of each coefficient $c_{(u,v)}$ by its corresponding quantifier $q^{\mathrm{QF}}_{(u,v)}$:

$$c'_{(u,v)} = \left\lfloor \frac{c_{(u,v)}}{q^{\mathrm{QF}}_{(u,v)}} \right\rfloor,\ u,v \in \{0,7\}. \tag{4}$$

The integer division produces a loss of precision which impacts the dequantization step. It is the inverse function that returns the input for the IDCT:

$$\hat{c}_{(u,v)} = c'_{(u,v)} \times q^{\mathrm{QF}}_{(u,v)},\ u,v \in \{0,7\}. \tag{5}$$

The quantization table is saved in the JFIF header and the quantified DCT blocks are compressed using entropy encoding.

JPEG entropy coding

The quantization produces small integers and zeros AC coefficients. As DC coefficients contain a lot of image energy and are very predictable, they are coded using a one dimensional predictor. AC coefficients are encoded following a zig zag order. The zig zag order is used to classify frequencies from the lowest to the higher. In zig zag order blocks often end with zeros since high frequencies are more quantified. Thus, after the last non-zero coefficient, an End Of Block (EOB) symbol is added to the MCU. The entropy encoder is illustrated in Fig. 2 which presents the main steps.

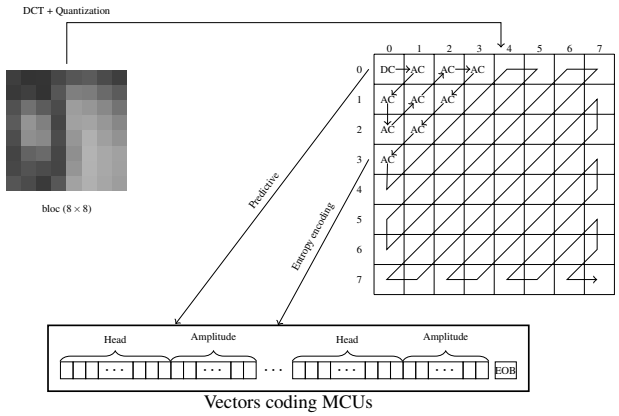


Figure 2: Entropy encoder.

The quantized DCT coefficients are coded by a pair (HEAD, AMPLITUDE) of binary vector. The variable HEAD contains run-length coding for zero values and run-size values, the Huffman code of non-zeros coefficients. Meanwhile the HEAD of DC coefficients is only made up by its size. The amplitude part corresponds to the amplitude of non-zero AC coefficients, or in the case of DC is the difference between two neighbouring DC coefficients. The i^{th} pixel block is then represented by a MCU as:

$$B^i = [(H^i_0, A^i_0), \ldots, (H^i_j, A^i_j), \ldots, (H^i_k, A^i_k)], \tag{6}$$

where, $0 \leq k \leq 64$ is the number of element in the MCU. The sequence of MCUs is placed after the header in JFIF bitstream.

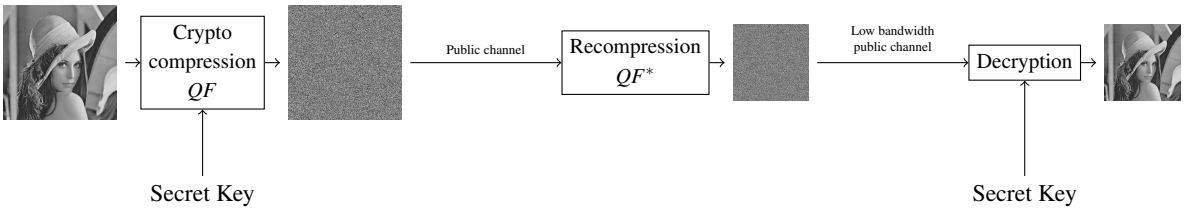


Figure 3: Crypto-compressed JPEG image is recompressed with $QF^\star < QF$

JPEG encryption

Several JPEG encryption techniques have been proposed because they are format compliant, they do not increase much the size, they hide content or partially hide content and they are secure. In order to preserve privacy and security, images should be encrypted during transmission or storage. Recently number of approaches have been proposed in order to encrypt JPEG compressed images. The encryption should be format compliant to preserve JPEG structure and be viewable. The main challenge is to not increase the file size, which is against compression rules. Authors have proposed size preserving JPEG encryption, which aims to keep the same size [18, 8, 12] or limit the size expansion [9, 10]. First partial encryption methods using sign encryption have been shown unsecure by Said [15]. Rodrigues *et al.* [14] propose a partial encryption which is applied selectively on regions of interest of the image such as faces. This method which relies on XOR operation with the AES algorithm, performs the compression and the encryption in the same process. Partial encryption is sufficient to keep hiding sensitive information such as text or faces. Moreover, it has the advantage to not change the size of the encrypted file. It can be used in the case of selective encryption as done in [11, 12]. A reversible watermarking method in encrypted domain is proposed in [13]. This method relies on XOR operation but for more visual masking author encrypt also quantization table. Blocks and coefficients scrambling is used in [9, 18, 8, 10]. Simple scrambling methods tend to increase the size if there is no verification of the run-length for example. In addition, simple scrambling methods can be attacked [7]. Using the non-zero-counting attack, authors extracts binary image from an encrypted one. They also propose a secure method based on full inter-block shuffle (FIBS). In [18], authors propose a partial encryption which preserves the size. Moreover, their method is claimed secure against attacks.

Recompression of crypto-compressed JPEG image

In this section, we develop our proposed method to recompress a crypto-compressed JPEG image. Assume Alice uses crypto-compression in order to protect her image against malicious user, and she allows format compliant recompression. It does not require specific viewer and it can allow partial visualization in case of selective encryption. Alice does not need to know the state of the bandwidth, she can crypto-compress her image in order to let the network provider recompress her image without the encryption key. The proposed scenario is presented in Fig. 3. First, the image is JPEG crypto-compressed using a quality factor $QF \in [1,100]$. Through the network the image is recompressed to a lower quality factor $QF^\star < QF$. The advantage of our method is that the decryption process before recompression is not necessary and then it is possible to recompress it without knowing the secret key. Moreover, this method does not need to decompress JPEG image and the recompression processing is done into the JPEG bitstream. The main idea of the recompression is for each coefficient code, to remove its last amplitude bit which corresponds to a division by two in base-10. In order to make a correct decoding, the quantization table has to be multiply by two. This method is compatible with the presented encryption because we can locate the removing part and shift the ciphering sequence.

First, we present the JPEG bitstream encryption while preserving its format and its size. Then, we explain the recompression steps of crypto-compressed image. Finally, we analyze the behavior of the quality factor during the recompression.

Preserving size JPEG crypto-compression

Starting from a raw image, we propose to crypto-compress the image into a JPEG format compliant bitstream. Since our method is designed to compress data, we use an encryption technique which does not increase the size of the bitstream compared to a simple JPEG compression. In this project, we propose to use the method presented by [12]. After defining the appropriate luminance quantization table, the method follows the JPEG compression steps. We propose to encrypt the amplitude part of the non-zero AC coefficients *i.e.* the concatenation of the amplitude of each coefficient of each block $[A_0^i,\ldots,A_k^i,\ldots,A_0^n,\ldots,A_k^n]$, where n is the number of blocks. The amplitude sequence is denoted $A = [a_0,\ldots,a_l]$ where l is the number of amplitude bits. A standard stream cipher function is used to generate a pseudo-random sequence $E = [e_0,\ldots,e_l]$ from a secret key. This sequence is XORed with the incoming plaintext, in order to produce the ciphered sequence $\widetilde{A} = [\widetilde{a}_0,\ldots,\widetilde{a}_l]$:

$$\widetilde{a}_i = a_i \oplus e_i,\ i \in [0,l], \tag{7}$$

where $\oplus$ denoted the XOR operator. The encrypted sequence is substituted to the amplitudes of the original bitstream. A coefficient $c_{(u,v)}$ is coded by the pair (H_k^i, A_k^i), where i is the block number and $k = u \times 8 + v$. This coefficient is encrypted using Eq. (7), to its new code $(H_k^i, \widetilde{A}_k^i)$ with the same number of bits. Fig. 4.b illustrates the result of a crypto-compression of the lena raw image presented in Fig. 4.a. The encryption is applied only on the non-zeros AC coefficients.

(a) (b)

Figure 4: a) Raw image: 256 KB, b) crypto-compressed JPEG image with quality factor set to 75: 35 KB, PSNR: 20.31 dB.

The encrypted bitstream can be decoded by a standard viewer but the content is partially masked, as presented in Fig. 4.b. There exists other preserving size encryption which are not suitable for recompression. Nevertheless, applying a full inter-block shuffle (FIBS) [7], in addition, allows to hide the content and add security. Although it increases the size of encrypted image, the additional cost remains low.

JPEG Recompression

The recompression is applied on the encrypted JPEG bitstream, which is the list of MCUs. They contains clear DC quantified coefficient and the encrypted AC quantified coefficients, noted $c'_{(u,v)}$, $u,v \in [0,7]$ encoded by the pair (H^i_k, A^i_k) for DC coefficient and $(H^i_k, \widetilde{A}^i_k)$, where i is the MCU number and $k = u \times 8 + v$. The binary representation of A^i_k is noted $[a_0, \ldots, a_l]$ where $l+1$ is its binary size. First, each coefficient is divided by two:

$$c^{\star}_{(u,v)} = \left\lfloor \frac{c'_{(u,v)}}{2} \right\rfloor, \tag{8}$$

which corresponds to removing the last bit a_l of the binary representation of A^i_k. If it amplitude is set to zero, it is coded in the run-lenght code of another coefficient, else the new value $\overset{\star}{A}{}^i_k$ contains the bits $[a_0, \ldots, a_{l-1}]$. Then, its corresponding head part H^i_k is adapted in function of the new amplitude size and in function of the number of preceding zeros if it changes. The head part is the Huffman code of the concatenation of the binary representation of the run-length and size values for AC coefficients, only size for DC coefficients. Removing the last bit implies reducing the binary size by one for all coefficients. Finally, $c^{\star}_{(u,v)}$ is encoded by $(\overset{\star}{H}{}^i_k, \overset{\star}{A}{}^i_k)$. The adapted quantization table $T^{\star}$ is given as:

$$q^{\star}_{(u,v)} = q^{\mathrm{QF}}_{(u,v)} \times 2. \tag{9}$$

The main advantage of this method is that it localizes the removed bits and the decryption is still possible. Then, the head part of an encoded coefficient has to be updated. The size part is decremented by one since we remove one bit. Moreover, if there is no more zeros before the coefficient, the runlength does not change. This method is compatible with the used encryption method because we can locate the removed part and shift the ciphering sequence. There is an exception for coefficients whose size is 1 bit. These coefficients become 0 after recompression and are included in entropy coding. Thus, in order to solve this problem, we propose to not encrypt these kinds of coefficients during the first crypto-compression. The quality impact is very low since the encryption of these coefficient does not change the value in 50% of cases in average. We are limited to one recompression in this case. Nevertheless, if coefficients of size lower than the number of recompression are not encrypted, the decryption is possible. Applying more recompression is possible at the expense of the encryption quality. We assume that a single recompression is sufficient in many cases. Moreover, the recompression can be applied on JPEG scrambled encryption method such as full inter-block shuffle (FIBS) [7]. Indeed, shuffling AC coefficients has no impact on their amplitude code, but it can change the entropy coding. In the experimental results, we propose then to analyze encryption effect on the compression rate. An example of the recompression in encrypted domain is illustrated in Fig. 5.

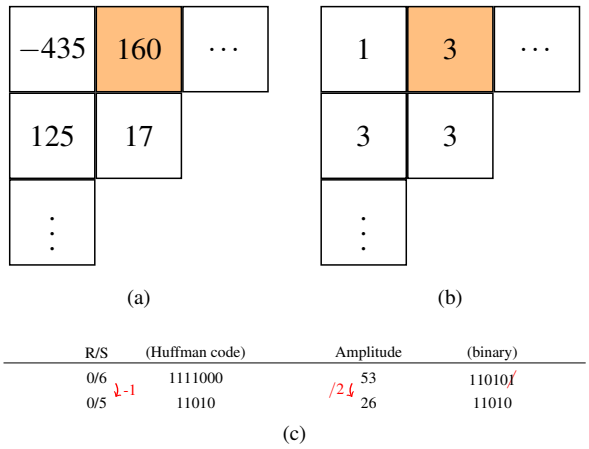


Figure 5: a) DCT Coefficients, b) Quantization table, c) Entropic coding

Fig. 5.a presents the DC and low frequency coefficient of a MUC. Suppose that an AC coefficient $c_{(0,1)} = 160$ quantified by $q^{\mathrm{QF}}_{(0,1)} = 3$, thus its new value is $c'_{(0,1)} = 53$. Then, suppose that $c'_{(0,1)} = 53$ is preceded by none 0 in zigzag order. Then, its run-length/size code is: $0/6$. A standard Huffman table gives the code: 1111000, thus the binary code for this coefficient is: 1111000110101. The coefficient is coded on 13 bits. The recompression is done by dividing by 2 the value, *i.e.* by removing the last bit. Thus, the run-length/size code is: $0/5$, assuming there is still no 0 before. And finally, we get 10 bits: 1101011010, coding the coefficient $c^{\star}_{(0,1)} = 26$. The quantization table is multiplied by two, so the dequantized value $\hat{c}_{(0,1)} = 26 \times (3 \times 2) = 156$.

A JPEG recompressed encrypted image is still viewable and the decryption can be perform with the key if we know there was a recompression. Indeed, we add in the JPEG comment part, 8 bits coding 0 or 1 that correspond respectively to no recompression and recompression. This recompression can be applied on any kind of JPEG encoded image encrypted with a compatible method such as the presented JPEG crypto-compression.

Quality factor analysis

In this section, we analyze how to evaluate objectively the compression quality of the produced recompressed crypto-compressed image. Indeed, after recompression, the image qual-

ity cannot be calculated without deciphering. The quality factor proposed by the IJG provides a good index of the compression quality. The final JPEG format file size is studied in Section Results. The recompression method produces quantization tables which are not standard. Furthermore, generally, there is no information in the JFIF header that contains the quantization factor. The problem is then to estimate the quality factor from a quantization table, this is illustrated in Fig. 6, where an original quantization table T_{80}, presented in Fig. 6.a, is multiplied by two to compensate the recompression. The image was compressed with a quality factor $QF = 80$ which is not know but can be retrieved. Nevertheless, the quality factor corresponding of the new quantization table $T^\star$ presented in Fig. 6.b, is not straightforward.

	0	1	2	3	4	5	6	7
0	6	4	4	6	10	16	20	24
1	5	5	6	8	10	23	24	22
2	6	5	6	10	16	23	28	22
3	6	7	9	12	20	35	32	25
4	7	9	15	22	27	44	41	31
5	10	14	22	26	32	42	45	37
6	20	26	31	35	41	48	48	40
7	29	37	38	39	45	40	41	40

(a)

	0	1	2	3	4	5	6	7
0	12	8	8	12	20	32	40	48
1	10	10	12	16	20	46	48	44
2	12	10	12	20	32	46	56	44
3	12	14	18	24	40	70	64	50
4	14	18	30	44	54	88	82	62
5	20	28	44	52	64	84	90	74
6	40	52	62	70	82	96	96	80
7	58	74	76	78	90	80	82	80

(b)

Figure 6: a) T_{80}, b) $T^\star$ for a recompression.

Thus Chandra and Ellis [2] proposed a predictor to measure the IJG equivalent of the JPEG quality factor for a JPEG image. Nevertheless, their predictor is not analyzed and there are no proof of correctness. We propose to estimate the quality factor of the new quantization tables from the definition of IJG for recompressed image. In order to evaluate the quantization factor of the produced tables, we invert Eq. (2). Thus, the mean inverse function is computed with the two possible equations:

$$f_1(T^\star) = \left\lfloor \frac{1}{n} \sum_{u=0}^{7} \sum_{v=0}^{7} 100 - \frac{50 \cdot p^{\mathrm{QF}}_{(u,v)} - 25}{q^{50}_{(u,v)}} \right\rfloor \tag{10}$$

$$f_2(T^\star) = \left\lfloor \frac{1}{n} \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{5000 \cdot q^{50}_{(u,v)}}{100 \cdot p^{\mathrm{QF}}_{(u,v)} - 50} \right\rfloor \tag{11}$$

Finally the quality factor is given by:

$$f(T^\star) = \begin{cases} f_2(T^\star), & \text{if } f_2(T^\star) \leq 50 \\ f_1(T^\star), & \text{otherwise.} \end{cases} \tag{12}$$

The quality factor estimation is denoted $QF^\star = f(T^\star)$ and $T^\star = T_{\mathrm{QF}^\star}$. The inversion works perfectly, if the Eq. (3) is not considered. The range value limitation implies that values smaller than 1 and greater than 255 are lost. Extreme cases are then defined for by two quantization tables T_{QF^-} and T_{QF^+}, where:

- $p^{\mathrm{QF}^-}_{(u,v)} = 1,\ \forall u,v \in [0,7]$,
- $p^{\mathrm{QF}^+}_{(u,v)} = 255,\ \forall u,v \in [0,7]$.

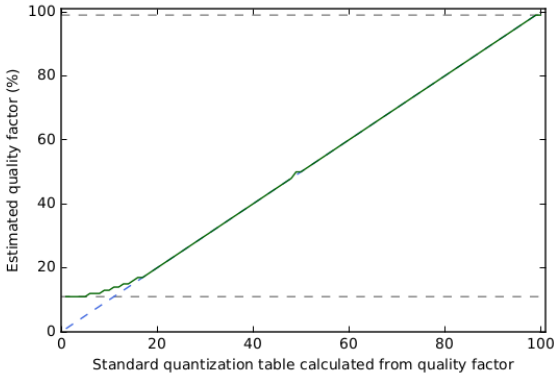


Figure 7: Quality factor estimation from quantization table.

Thus, the range of the proposed quality factor estimation is $[f(T_{\mathrm{QF}^-}), f(T_{\mathrm{QF}^+})] = [11, 99]$. The quality factor estimation function is plotted in Fig.7, for standard quantization table. We can notice the divergence of the function for quality factors below than 20 and for the value 100. To evaluate this divergence, we propose to evaluate the quality factor $QF^\star$ of standard quantization table T_{QF}, then we generate the standard quantization table $T_{\mathrm{QF}^\star}$ with the evaluated quality factor. If the quality factor is correctly estimated then the $L2$-distance between T_{QF} and $T_{\mathrm{QF}^\star}$ is null. Results of $L2$-distance divergence are presented in Fig. 8 for all standard quantization table with $QF \in [1, 100]$. It is observed that

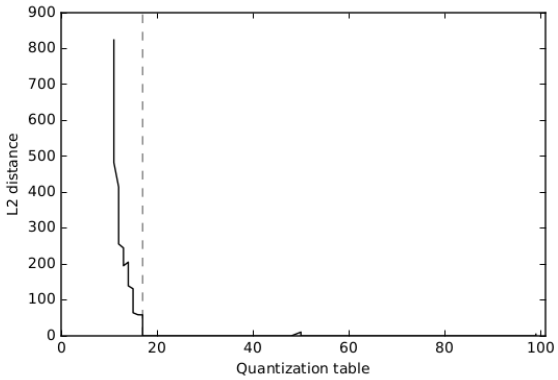


Figure 8: $L2$-distance between standard quantization table and the estimated quantization table.

while the estimated $QF^\star$ diverges the distance between standard quantization table and the estimated quantization table increases. This study is done to evaluate the compression quality after the recompression, so we evaluate the $L2$-distance between the recompressed standard quantization table for which we evaluate a quality factor $QF^\star$ and the estimated quantization table given by $QF^\star$. Results are presented in Fig. 9. From this figure, we note that the divergence starts for an estimated quality factor around 34 to the limit at 11. We also notice a range where the function has a sawtooth shape, which is due to integer rounding errors. Nevertheless, we can well estimate the recompressed image quality for an estimated $QF^\star$ greater than 34. The estimated correspondance between a standard table quality factor and the new quality factor value after recompression is presented in Fig. 10.

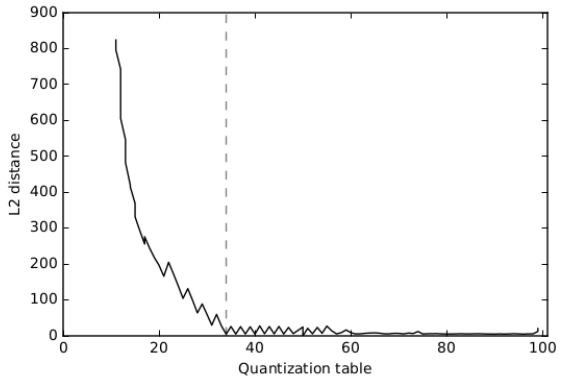


Figure 9: $L2$-distance between standard quantization table and the estimated quantization table after recompression.

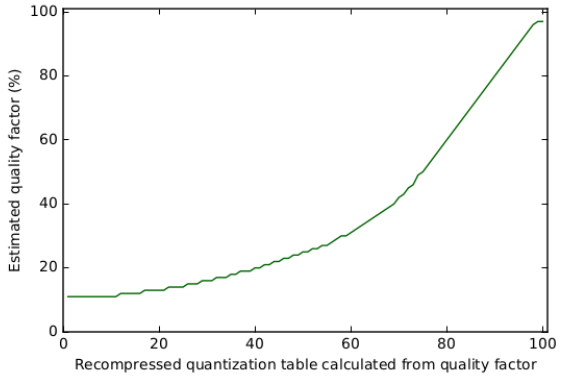


Figure 10: $L2$-distance between standard quantization table and the estimated quantization table after recompression.

This analysis allows us to guess the recompression quality even before the crypto-compression of the image and to choose wisely an appropriate quality factor. Moreover, this analysis is used to assess the image recompression quality for experimental results.

Experimental results

In this study we evaluate the image quality with the PSNR (Peak Signal-to-Noise Ratio) between the compressed images and their original corresponding raw images. We also evaluate the compression rate presented in bpp (bit per pixel).

(a) (b) (c)

Figure 11: a) JPEG crypto-compressed image with $QF = 75$ (46 KB), b) The recompressed image with $QF^\star = 50$ (26 KB), c) Decryption of the recompressed crypto-compressed image (26 KB).

Fig. 11.a presents the result of the JPEG crypto-compression, of the raw image Barbara (256 KB) with a quality factor QF of 75%. The compression rate is about 0.18 bpp and the PSNR is 17.9 dB. Note that the decryption of this image produces a good quality image with a PSNR of 36.3 dB. In Fig. 11.b the image of Fig. 11.a, is recompressed with our proposed method to an approximate quality factor of 50%, achieving a compression rate of 0.10 bpp. The PSNR of the recomressed crypto-image is almost the same around 17.9 dB. Finally, the recompressed crypto-image can be decrypted with the good key as shown in Fig. 11.c. The image is correctly decrypted with a PSNR of 31.9 dB.

Our experimental results have been applied on 117 standard gray-scale images from different databases[1]. Each image is crypto-compressed, then recompressed and finally decrypted. Table. 1 presents in the first row the quality factors used for crytpo-compression. The corresponding approximate quality factors calculated after recompression are presented in the second row.

Table 1: Sample of quality factors used in our experiments and their corresponding approximate quality factors after recompression.

$QF =$	100	95	90	75	50	25	15
$QF^\star \simeq$	97	90	80	50	25	14	12

Fig. 12 presents the compression rate as a function of the PSNR. It presents the average of values acquired on the 117 images of the base for each quality factor and their corresponding approximate quality factor after recompression. This method is designed to work in encrypted domain but it should be efficient in terms of compression rate and PSNR. Thus, Fig. 13 presents the

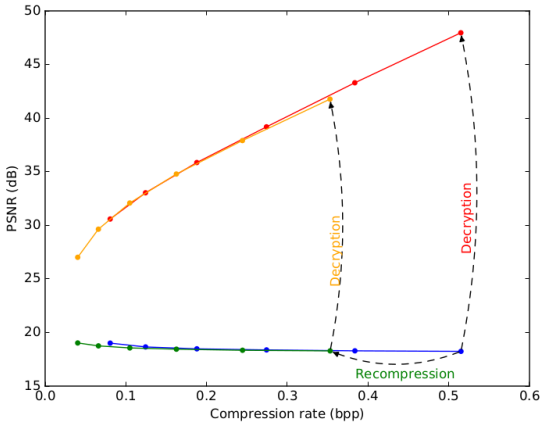


Figure 12: Average compression rate for 117 images as a function of the average PSNR, in blue the proposed JPEG crypto-compression, in green the recompression of the proposed crypto-compression, in red decryption of the proposed crypto-compression and in orange decryption of the recompressed images.

[1] http://decsai.ugr.es/cvg/index2.php

comparison between the standard JPEG compression and the decryption of the recompressed JPEG crypto-compression in terms of PSNR and compression ratio. The estimated quality factor calculated after recompression is used as standard quality factor (*i.e.* quantization table is generated from standard table) for standard JPEG compression. It can be seen that the curve of the decryp-

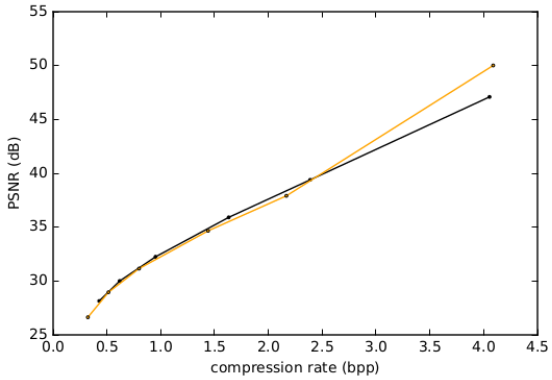


Figure 13: Average compression rate for 117 images as a function of the average PSNR, in black classical JPEG compression, in orange the decryption of the recompressed JPEG crypto-compressed images.

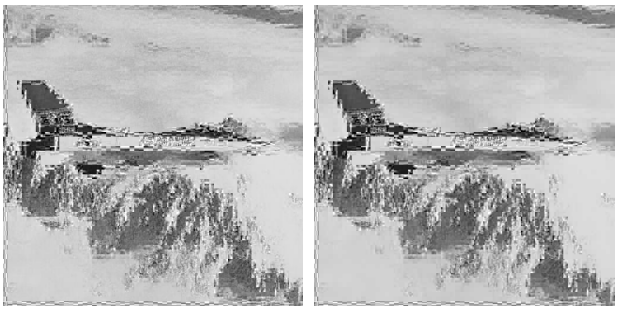
tion recompressed images, is very closed to the JPEG standard compression. It shows that the modified quantization table fit almost fit the JPEG compression with standard quantization table. However, the proposed recompression is similar to standard JPEG recompression which produces block artifacts [1], as illustrated in Fig. 14. Fig. 14.a presents these artifacts on the decryption

(a) (b)

Figure 14: a) Decryption of recompressed crypto-compressed image with $QF^\star = 25$, 32.9 dB, b) direct JPEG compression with $QF = 25$, 33.7 dB

of recompressed crypto-compressed image with $QF^\star = 25$. The recompressed image is obtained from the crypto-compressed image with $QF = 50$. Fig. 14.b present the direct JPEG compression with $QF = 25$ from standard quantization table. These images have a similar PSNR but the recompressed image looks more noisy because of block artifacts which is visually disturbing. This kind of artifacts are more frequent for certain quality factors as explained by [1].

The main objective of recompression of crypto-compressed image is to reduce the size while preserving the security of the encrypted visual information. An example is presented in Fig. 15, first the image is crypto-compressed from the raw image with $QF = 75$, the file size is 32.6 KB. Then, the image is recompressed with an estimated $QF^\star = 50$, it has a size of 18.5 KB, which is a gain of 43% in size. The crypto-compression qual-



(a) (b)

Figure 15: a) Crypto-compressed image with $QF = 75$, 32.6 KB, b) recompressed image with $QF^\star = 50$, 18.5 KB

ity factor can be chosen to take into account the recompression. Indeed, knowing the system constraints can be usefull in order to choose the best trade-off between recompressed file size and its visual quality. Nevertheless, it is absolutely not necessary to know the quality factor of the crypto-compressed image for applying the recompression. We experimentally compute the mean gain in size of the 117 images of the base for the quality factor sample. Results are presented in Fig. 16. It can be seen that

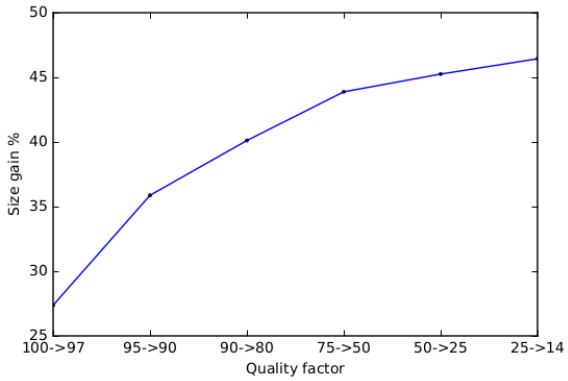


Figure 16: Average gain in file size in percentage for 117 for the quality factor sample.

the recompression gain in size is always superior of a quarter of the crypto-compressed image size. Moreover, for more standard applications, such as a crypto-compression with $QF = 75$, the recompression rises to 42% of file size.

Conclusion

In this paper we propose an efficient method to recompress JPEG crypto compressed images. The preservation of the size of the JPEG crypto-compression allows us to keep the size comparable to a simple JPEG compression. The recompression method efficiently reduces the JPEG crypto-compressed image size. Moreover, the compression rate is very close to the standard JPEG one.

In this paper, we focused on the luminance channel and thus we presented results on gray scale image. The method can be applied on color image on its luminance component and extended to chrominance channels. In future work, we will consider different scenarios such as selective, partial or even full encryption. Furthermore, we are investigating the best quality factor for crypto-compression in order to produce few artifacts and thus, have the best image quality after compression while maintaining an efficient size gain.

References

[1] H. H. Bauschke, C. H. Hamilton, M. S. Macklem, J. S. McMichael, and N. R. Swart. Recompression of JPEG images by requantization. IEEE Transactions on Image Processing, 12(7):843–849, 2003.

[2] S. Chandra and C. S. Ellis. JPEG Compression Metric as a Quality Aware Image Transcoding. 2nd USENIX Symposium on Internet Technologies & Systems (USITS'99), 1999.

[3] E. Hamilton. JPEG file interchange format. C-Cube Microsystems, 1992.

[4] C. Y. Hsu, C. S. Lu, and S. C. Pei. Image feature extraction in encrypted domain with privacy-preserving SIFT. IEEE Transactions on Image Processing, 21(11):4593–4607, 2012.

[5] P. Korshunov and T. Ebrahimi. Scrambling-based tool for secure protection of JPEG images. In IEEE International Conference on Image Processing, pages 3423–3425, 2014.

[6] R. Lazzeretti and M. Barni. Lossless compression of encrypted grey-level and color images. In 16th European Signal Processing Conference, pages 1–5, 2008.

[7] W. Li and Y. Yuan. A leak and its remedy in JPEG image encryption. Int. J. Comput. Math., 84(9):1367–1378, 2007.

[8] K. Minemura, Z. Moayed, K. Wong, X. Qi, and K. Tanaka. JPEG image scrambling without expansion in bitstream size. In IEEE International Conference on Image Processing, pages 261–264, 2012.

[9] X. Niu, C. Zhou, J. Ding, and B. Yang. JPEG encryption with file size preservation. In IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pages 308–311, 2008.

[10] S. Ong, K. Wong, X. Qi, and K. Tanaka. Beyond formatcompliant encryption for JPEG image. Signal Processing: Image Communication, 31:47 – 60, 2015.

[11] M. Pinto, W. Puech, and G. Subsol. Protection of JPEG compressed e-comics by selective encryption. In IEEE International Conference on Image Processing, pages 4588–4592, 2013.

[12] W. Puech, A. G. Bors, and J. M. Rodrigues. Advanced Color Image Processing and Analysis, chapter Protection of Colour Images by Selective Encryption, pages 397–421. Springer New York, 2013.

[13] Z. Qian, X. Zhang, and S. Wang. Reversible data hiding in encrypted JPEG bitstream. IEEE Transactions on Multimedia, 16(5):1486–1491, 2014.

[14] J. M. Rodrigues, W. Puech, and A. G. Bors. Selective encryption of human skin in JPEG images. In IEEE International Conference on Image Processing, pages 1981–1984, 2006.

[15] A. Said. Measuring the strength of partial encryption schemes. In IEEE International Conference on Image Processing, 2:II–1126–9, 2005.

[16] D. Schonberg, S. Draper, and K. Ramchandran. On compression of encrypted images. In IEEE International Conference on Image Processing, pages 269–272, 2006.

[17] International Telecommunication Union. CCITT recommendation T.81, Information Technology-Digital Compression and Coding of Continuous-Tone Still Images Requirements and Guidelines, 1992.

[18] A. Unterweger and A. Uhl. Length-preserving bit-streambased JPEG encryption. In ACM Proceedings of the on Multimedia and Security, MM& Sec '12, pages 85–90, 2012.

[19] M. Van Droogenbroeck and R. Benedett. Techniques for a selective encryption of uncompressed and compressed images. In Advanced Concepts for Intelligent Vision Systems, pages 90–97, 2002.

[20] G. K. Wallace. The JPEG still picture compression standard. IEEE Transactions on Consumer Electronics, 38(1):xviii–xxxiv, 1992.

[21] G. Zhang, S. Liu, F. Jiang, D. Zhao, and W. Gao. An improved image compression scheme with an adaptive parameters set in encrypted domain. In Visual Communications and Image Processing, pages 1–6, 2013.

Author Biography

Vincent ITIER *received the M.S degree in Computer Science from the University of Montpellier, France, in 2012 and the Ph.D. degree in Computer Science from the University of Montpellier, France, in 2015. Currently, he is Post-Doc research at Telecom Lille, France. His work has focused on the development of 3D mesh data-hiding applications, JPEG compression and encryption.*

William PUECH *received a diploma in Electrical Engineering from the University of Montpellier, France, in 1991 and a Ph.D. Degree in Signal-Image-Speech from the Polytechnic National Institute of Grenoble, France in 1997. He started his research activities in image processing and computer vision. He served as a Visiting Research Associate to the University of Thessaloniki, Greece. From 1997 to 2000, he has been an Assistant Professor at the University of Toulon, France, with research interests including methods of active contours applied to medical images sequences. Between 2000 and 2008, he has been Associate Professor and since 2009, he is full Professor in image processing at the University of Montpellier, France. He works in the LIRMM Laboratory (Laboratory of Computer Science, Robotic and Microelectronic of Montpellier). His current interests are in the areas of protection of visual data (images, videos and 3D objects) for safe transfer by combining watermarking, data hiding, compression and cryptography. He has applications on medical images, cultural heritage and video surveillance. He is the head of the ICAR team (Image & Interaction) and he has published 39 journal papers, 16 book chapters and more than 100 conference papers. W. Puech is associate editor of J. of Advances in Signal Processing, Springer, Signal Processing: Image Communications, Elsevier, Signal Processing, Elsevier and IEEE Transactions on Dependable and Secure Computing. He is reviewer for more than 15 journals (IEEE Trans. on Image Processing, IEEE Trans. on Multimedia, IEEE Trans. on Circuits and Systems for Video Technology, IEEE Trans. on Information Forensic and Security, Signal Processing: Image Communication, Multimedia Tools and Applications ...) and for more than 10 conferences (IEEE ICIP, IEEE ICASSP, EUSIPCO, ...).*