This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2019.2894520, IEEE Transactions on Circuits and Systems for Video Technology

Recompression of JPEG crypto-compressed images without a key

Vincent Itier, Pauline Puteaux *Student Member, IEEE* and William Puech, *Senior Member, IEEE*

***Abstract*—The rising popularity of social networks and cloud computing has greatly increased number of JPEG compressed image exchanges. In this context, the security of the transmission channel and/or the cloud storage can be susceptible to privacy leaks. Selective encryption is an efficient tool to mask image content and to protect confidentiality while remaining format-compliant. However, image processing in the encrypted domain is not a trivial task. In this work, we present a JPEG crypto-compression method which allows us to recompress a JPEG crypto-compressed image several times, without any information about the secret key or the original image content. Indeed, using the proposed method in this paper, each recompression can be done directly on the JPEG bitstream by removing the last bit of the code representation of each non-zero coefficient, adapting the entropic code part, and slightly modifying the quantization table. This method is efficient to recompress JPEG crypto-compressed images in terms of ratio compression. Moreover, the decryption of the recompressed image produces an image with a very similar visual quality when compared to the original image, according to the obtained results.**

***Index Terms*—JPEG compression, selective image encryption, image security, signal processing in the encrypted domain, recompression.**

I. Introduction

In the last few years, the growing popularity of cloud storage and network sharing has led to the demand for greater security and privacy of personal data [1]. In fact, during the transmission and/or the storage of multimedia data, confidentiality, authentication and integrity are constantly being threatened by illegal activities, such as hacking, copying or malicious use of information. Securing the access to the file is not enough. The content should be protected itself, and this can be implemented by encryption for example. Furthermore, the rapid growth of network usage has led to greater needs in bandwidth which is limited. The most popular image compression standard is JPEG [2]. In order to exploit both the efficient compression and encryption, format-compliant methods are designed to produce content compatible with format specifications. There are format-compliant JPEG encryption methods which can be used in this context. The authors have proposed size preserving JPEG encryption [3]–[5] or, have limited the expansion of the size [6], [7]. Partial encryption methods using sign encryption have been exposed as insecure by Said [8]. In the work of Puech *et al.* [5], a partial encryption is applied selectively on automatically detected 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 [9]. Moreover, it has the advantage of not changing the size of the encrypted file. Blocks and coefficients scrambling is used in [3], [4], [6], [7]. Simple scrambling methods tend to increase the size if there is no verification of the run-length for example. Inter-block shuffle and non-zero AC scrambling methods have been exposed as liable to sketch attacks [10], [11]. Other authors propose a specific format for compression of encrypted images [12]–[14], these are limited due to the removal of the redundancy by the encryption. Moreover, encrypted images should be compatible with most viewers, social networks, cloud storage *i.e.* format-compliant.

V. Itier, P. Puteaux and W. Puech are with the Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier, Centre National de la Recherche Scientifique, Univ. Montpellier, Montpellier 34095, France (e-mail: vincent.itier@lirmm.fr; pauline.puteaux@lirmm.fr; william.puech@lirmm.fr).

JPEG crypto-compressed images should be recompressible with the aim to be adapted to limited bandwidth or storage, for example. Usually, when a bandwidth is limited, a network node can perform a recompression of a heavy JPEG file. Classic JPEG recompression consists of decoding the JPEG file and applying a JPEG compression to the decoded pixels. As shown by Chan, some artifacts - grainy effect and loss of sharpness - appear on the image after the second compression to a lower quality factor [15]. The same author also remarks that these artifacts do not appear if the compression to the lower quality factor is directly applied. In keeping with this work, Bauschke *et al.* explain that if a JPEG image with a quality factor of 75% is classically recompressed with a quality factor of 50%, the grainy effect appears and alters perceptually the image. However, this is not the case with a smaller quality factor of 48%: the quality rating scale is thus not perceptually monotone [16]. They also propose an analysis of the problem and a recompression algorithm in order to prevent it. It is then possible to investigate the effect of multiple JPEG compressions for forensics applications [17]. Lewis and Kuhn [18] defined four main classes of recompressors: a recompressor can be exact, complete, stable or naive. Naive recompression consists of applying a standard compression on decompressed data. It produces a non-monotone quality of recompressed images as it has been shown in [15], [16]. Therefore, authors have proposed exact recompressors, that produce the same output as the input decompressed data. Complete recompressors focus on generating a set of equivalent inputs, while stable recompressors can localize loss of information during recompression.

The problem lies in applying recompression in the encrypted domain. In fact, direct JPEG recompression of crypto-compressed images does not allow decryption. Thus, the method proposed in [16] for example, cannot be applied in the encrypted domain. A potential solution would be to share the encryption key with the service provider. Thanks to the key, it can perform the decryption of the crypto-compressed image, recompress the reconstructed image in clear, and finally, encrypt it with the same encryption key as before. Nevertheless, this scheme is insecure and may be susceptible to leaks, because the service provider has access to the original image content.

As a solution to solve this problem, in this paper, we present a method of recompression of crypto-compressed JPEG images. First, we propose a new JPEG crypto-compression method based on [5], but which is robust to multiple recompressions. Indeed, it is not possible to apply a recompression in the encrypted domain by using the method in [5]. In our scheme, a JPEG compression and an encryption of the sorted non-zero quantized DCT coefficients are jointly performed during the Huffman coding stage. This encryption procedure preserves both the JPEG format and the compression rate, which is exactly the same compared to a simple JPEG compression of the same image. Next, the crypto-compressed image can be uploaded onto a cloud platform and, if necessary, it can be recompressed directly by the service provider, without any access to the clear image content or the encryption key. Moreover, the recompression method achieves a very good compression rate for the obtained recompressed crypto-compressed JPEG image and the decrypted recompressed crypto-compressed JPEG image is very similar to the original image.

The rest of this paper is organized as follows. Section II gives an overview of the JPEG algorithm and of related work on image crypto-compression. Then, the proposed method is described in detail, with example of application, in Section III. Experimental results and analysis are provided in Section IV. Finally, the conclusion is drawn in Section V.

II. Related work

A. JPEG compression

JPEG (Joint Photographic Experts Group) is the most popular method of lossy compression for digital images [2]. It has been standardized by the IJG (Independent JPEG Group). Moreover, in order to encapsulate images compressed with JPEG, the JFIF (JPEG File Interchange Format) is often used [19].

According to JPEG standard, a RGB image represented by three components red, green and blue, is first converted into luminance/chrominance space (YCrCb). Then, the two chrominance components may be subsampled. In fact, the human visual system (HVS) can see considerably more fine details in the luminance (Y component) of an image than in the chrominance (Cr and Cb components). Using this knowledge, in order to compress images more efficiently, it is possible to reduce the spatial resolution of the Cr and Cb components with a subsampling. After this step, each component is encoded separately, by applying the same transformations. First, they are decomposed into non-overlapping blocks of 8×8 pixels on which a DCT transformation is applied. After the DCT transformation, the frequency coefficients are floating values and a quantization operation is necessary to convert them into 8 bits integers and to reduce their range. This operation causes the loss of information in JPEG compression. The final step of JPEG compression is entropy coding, where the run-length coding algorithm (RLC) and then Huffman coding are performed. Section II-A1 to Section II-A3 give a detailed description of these last three steps.

1) DCT transformation: The obtained 8×8 pixels blocks of each component are transformed from the spatial to the frequency domain using the Discrete Cosine Transform (DCT):

$$F(u,v)=\frac{1}{4}C(u)C(v)\sum_{i=0}^{7}\sum_{j=0}^{7}p(i,j)\cos\left[\frac{(2i+1)u\pi}{16}\right]\cos\left[\frac{(2j+1)v\pi}{16}\right], \tag{1}$$

with $p(i,j),\ 0 \leq i,j < 8$ the pixels of the 8×8 block of the original image, $F(u,v),\ 0 \leq u,v < 8$ the computed DCT coefficients and $C(x) = \frac{1}{\sqrt{2}}$ for $x = 0$, $C(x) = 1$ for $x > 0$.

There are two types of DCT coefficients: the DC and the AC coefficients. The DC coefficient, $F(0,0)$, corresponds to the zero frequency and is relative to the average value of the block. The AC coefficients, $F(u,v)$, with $0 \leq u,v < 8$ and $(u,v) \neq (0,0)$, relate to the frequency information. Note that the more (u,v) is close to $(8,8)$, the more the frequencies are high and imperceptible for the HVS. Moreover, even if the pixels are integers, the DCT coefficients $F(u,v)$ are floating values.

2) JPEG quantization: In order to decrease the size and since each coefficient $F(u,v)$ is a floating value, a quantization is necessary. From a quality factor $\text{QF} \in [1,100]$, a quantization table Q_{QF} is defined.

	0	1	2	3	4	5	6	7
0	16	11	10	16	24	40	51	61
1	12	12	14	19	26	58	60	55
2	14	13	16	24	40	57	69	56
3	14	17	22	29	51	87	80	62
4	18	22	37	56	68	109	103	77
5	24	35	55	64	81	104	113	92
6	49	64	78	87	103	121	120	101
7	72	92	95	98	112	100	103	99

Fig. 1: Standard luminance quantization table Q_{50}.

The IJG specifies a standard luminance quantization table Q_{50} for $\text{QF} = 50\%$, displayed in Fig. 1. From this table, the coefficients $q_{\text{QF}}(u,v)$ from each quantization table can be calculated:

$$q_{\text{QF}}(u,v) = \begin{cases} \left\lfloor \frac{q_{50}(u,v)\times\left(\frac{5000}{\text{QF}}\right)+50}{100} \right\rfloor, & \text{if QF} < 50, \\ \left\lfloor \frac{q_{50}(u,v)\times(200-2\text{QF})+50}{100} \right\rfloor, & \text{otherwise.} \end{cases} \tag{2}$$

In order to fulfill the IJG recommendation and for a full JPEG baseline compatibility, the coefficients $q_{\text{QF}}(u,v)$ have to remain integers, between 1 to 255. Under this constraint, we have:

$$q_{\text{QF}}(u,v) = \begin{cases} 1, & \text{if } q_{\text{QF}}(u,v) < 1, \\ 255, & \text{if } q_{\text{QF}}(u,v) > 255, \\ q_{\text{QF}}(u,v), & \text{otherwise.} \end{cases} \tag{3}$$

Note that for $\text{QF} = 100\%$, all the coefficients $q_{100}(u, v),\ 0 \leq u, v < 8$ are equal to 1. Even using this high quality, there is still a loss of information. The more the quality factor is small, the more the quantization coefficients are high and then, the degradation of the compressed image is more visible, due to the importance of the quantization step.

Since the quantization step is performed in order to encode DCT coefficients onto small integers, several coefficients are equal to zero after the quantization. This fact also increases the compression rate. Each DCT coefficient $F(u, v)$ is then divided by its corresponding quantization parameter $q_{\text{QF}}(u, v)$ from the table Q_{QF} to obtain the quantized coefficients $F'(u, v)$:

$$F'(u, v) = \left[\frac{F(u, v)}{q_{\text{QF}}(u, v)} \right]. \tag{4}$$

Note that this step is the main cause of image quality losses in JPEG compression since it is not reversible. During the decoding stage, the inverse function returns the input for the I-DCT:

$$\tilde{F}(u, v) = F'(u, v) \times q_{\text{QF}}(u, v). \tag{5}$$

The quantization table may be saved in the JFIF header and the quantized DCT blocks are then compressed using entropy coding.

3) JPEG entropy coding: After the quantization step, the quantized DCT coefficients are scanned in a zigzag order onto a vector, called Minimum Code Unit (MCU), according to their increasing spatial frequency. Using this method, blocks often end up with zeros since high frequency are more quantized. After the last non-zero coefficient, an End Of Block (EOB) symbol is added to the MCU. For each quantized DC coefficient, the difference with the quantized DC coefficient from the previous adjacent blocks is computed in order to calculate a prediction error. Then, this prediction error is encoded as the amplitude value $A_{F'(u,v)}$ of the quantized DC coefficient. The head $H_{F'(u,v)}$ of this coefficient contains the number of bits to represent this amplitude, *i.e.* the size parameter. For the quantized AC coefficients, a run-length coding (RLC) algorithm is applied to compress the consecutive coefficients equal to zero. On one hand, the value of each non-zero quantized AC is then encoded as the amplitude value $A_{F'(u,v)}$. On the other hand, the head $H_{F'(u,v)}$ of these coefficients is composed of the run-length computed previously and the amplitude size parameter. Finally, the head parameter of each quantized DCT coefficient is encoded using the Huffman algorithm. The sequence of MCU is then placed after the header in JFIF bitstream.

B. Image encryption

The aim of encryption is to guarantee data privacy and visual confidentiality of an original image. In these approaches, security is ensured by randomizing – selectively, partially or completely – the content of a clear image, by using a secret key. Cryptosystems can be symmetric, when the same key is used during the encryption and the decryption phases, like in AES or DES, or asymmetric, when there are public and private keys, like in RSA or in the Paillier cryptosystem. Moreover, in symmetric cryptography, data can be encrypted independently of the last operation or by utilizing previously encrypted content [20]. Although classic algorithms have been adapted, many other methods, such as scrambling techniques and chaos-based cryptography, have been specifically developed for image encryption in order to take into account image properties.

Scrambling techniques have also been designed in several papers. Efficient and easy to implement, their objective is to produce a non-intelligible image, by permuting the position of the pixels. Usman *et al.* suggested randomly permuting the rows and the columns of an image in order to break the correlation of the edges [21]. In [22], Premaratne *et al.* proposed a similar approach. Wright *et al.* proposed two scrambling techniques [23]. The first one consists of permuting the locations of the pixels within the blocks. In the second one, sub-blocks within the blocks are permuted and, after that, pixels in sub-blocks are shuffled.

Along with the rapid development of theory and application of chaos, a lot of image encryption schemes based on chaos theory have been presented. In most cases, in addition to a scrambling operation, the pixels values are substituted. Chaos-based image cryptosystems can be divided into two categories. In the first one, a pixel is considered as the smallest element [24]–[26] and, in the second, a pixel is composed by bits, on which bit-level operations are performed [27], [28]. Chen *et al.* employed a three-dimensional (3D) Arnold cat map [24] and Mao *et al.* used a 3D baker map [25] to shuffle the pixel positions during the substitution phase. Guan *et al.* applied the Arnold cat map to shuffle the positions of the image pixels in the spatial-domain and then, used the chaotic system of Chen and Ueta in [29] to modify the pixels values [26]. In order to reduce execution time, Xiang *et al.* suggested to encrypt only the four most significant bits of each pixel in their scheme described in [27]. Thus, this method is selective: the four last bits of each pixel remain in clear. In their paper [28], Zhu *et al.* proposed an image crypto-system where the Arnold cat map is used for bit-level permutation, this results in both pixel position and pixel value modifications. After that, the logistic map is employed for diffusion.

Furthermore, for acquisition, exchange and storage, compressed JPEG images are often used. Thus, many cryptosystems combining encryption and compression have been designed. They are known as crypto-compression algorithms.

C. Image crypto-compression

In the first methods of crypto-compression, encryption was done separately from the compression stage. However, the main problem with these approaches is that encryption significantly modifies the statistical characteristics of the image and, consequently, compression efficiency is severely reduced if the encryption is completed before. For this reason, in the last few years, there has been a growing research interest in studying how to encrypt JPEG compressed images in such a way that the encrypted data can still be represented in a meaningful format (format-compliant property). In this new kind of methods, JPEG compression and encryption are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2019.2894520, IEEE Transactions on Circuits and Systems for Video Technology

performed jointly. Three categories can be established: sign-bit encryption, DCT coefficient encryption and scrambling-based security methods.

Shi and Bhargava designed one of the first crypto-compression approaches allowing to perform encryption directly on the JPEG bitstream [30]. They proposed to encrypt sign-bits of both AC and DC coefficients (for DC coefficients, these are sign-bits of the differential values). A pseudo-random binary sequence is generated according to a secret key and encryption is then performed by XORing this sequence with the bitstream obtained by concatenation of all sign-bits. With this method, the JPEG structure is preserved (format-compliant property) and the compression rate is not modified. However, as shown by Said in [8], this scheme is insecure. In fact, due to format-compliance, a low complexity attack scheme can be designed. It is actually possible to guess encrypted bits using information from the rest of the data which is in clear.

In [31], Van Droogenbroeck and Benedett suggested encrypting the AC coefficients after DCT transformation, but not the DC coefficients, because they carry important visible information and are predictable. Puech and Rodrigues, in [32], proposed a selective encryption method for JPEG images, based on the encryption of both DC and AC coefficients. All of the DC coefficients and some AC coefficients of the lowest frequencies are concatenated to form a bitstream of 128 bits. This bitstream is encrypted using the AES algorithm. In [5], Puech *et al.* presented a method to perform encryption of some regions of interest (ROI) corresponding to the human skin. ROI are detected using the clear quantized DC coefficients of the two chrominance components Cr and Cb. Selective encryption is applied to blocks of the Y component, during the JPEG entropy coding phase. Using the AES algorithm in CFB mode, the quantized AC coefficients of the ROI are encrypted. This method is format-compliant and the JPEG crypto-compressed image has exactly the same size as with the standard JPEG algorithm. Shahid *et al.* designed a selective encryption technique for H.264/AVC video codec for CAVLC and CABAC [33]. Encryption is performed during the entropy coding stage and using the AES algorithm in CFB mode. In order to preserve the H.264/AVC format and the file size, encryption is only done on the CAVLC codewords and the CABAC binstrings. A survey of HEVC crypto-compression methods has also been proposed by Hamidouche *et al.* [34].

With the full inter-block shuffle (FIBS) method, proposed by Li and Yuan in [10], DC coefficients and same frequency coefficients are scrambled. This produces an unintelligible image. However, as all coefficients are scrambled, the compression performances of RLC are reduced. Scrambling coefficients of the same frequencies may change the header part of the code of a coefficient, because it may change run-length size. In case of DC coefficients, the efficiency of the predictive coding decreases and then much more bits are used to encode them. Minemura *et al.* proposed to scramble a JPEG image in order to encrypt it without causing bandwidth expansion [4]. They made some recommendations on the AC coefficients and built regions according to edge information induced by these coefficients. After that, DC coefficients with similar values are processed in groups. Actually, if only the AC coefficients are scrambled, the outline of the image is still revealed. Unterweger and Uhl described a crypto-compression approach based on three steps [3]. After permuting the order of the run-length coded symbols together with their corresponding coefficient values, bits of the coefficient values are scrambled, and finally, similar blocks are permuted.

Dufaux and Ebrahimi designed two different methods for privacy protection in video surveillance systems [35]. In the first one, the sign of some coefficients are pseudo-randomly flipped during the coding phase. In the second one, they pseudo-randomly scrambled some bits of the code-stream. In [36], Kurihara *et al.* designed an encryption-then-compression system where blocks are shuffled in the spatial domain. However, this encryption scheme can be broken using a jigsaw puzzle solver, as shown in [37]. In fact, authors considered the blocks of an encrypted image as pieces of a jigsaw puzzle: image decryption amounts to jigsaw puzzle assembly. Moreover, other encryption-then-compression schemes have also been described in [38]–[40].

Although most methods are based on discrete cosine transform (DCT), some other schemes exist and have been popular in the last few years. In fact, many crypto-compression methods are based on JPEG2000, as presented in the survey of the state-of-the-art methods performed by Engel *et al.* [41]. These schemes are also based on wavelet coefficient sign encryption [42], permutations [43], [44] or randomized arithmetic coding [45] for example.

III. Proposed method of recompression of a JPEG crypto-compressed image

In this section, we develop our proposed method of recompression of JPEG crypto-compressed images in the encrypted domain, without knowing the secret key. We first present a new method of crypto-compression which is robust to multiple recompressions. The JPEG compression and the encryption of the sorted non-zero quantized DCT coefficients are jointly performed during the Huffman coding stage. Then we describe our proposed method to recompress the crypto-compressed image directly in the encrypted domain, while maintaining the security level of the crypto-compression method. In Section III-A, we present an overview of the proposed method. The JPEG crypto-compression approach is described in Section III-B and our approach to recompress the crypto-compressed image is presented in Section III-C. In Section III-D, we explain the decoding of a crypto-compressed image after recompression, and finally, in Section III-E, we present a full application example of our method.

A. Overview of the proposed method

From an original image, the first step of JPEG consists of applying a color transformation, from RGB to YCrCb space, where the two chrominance components can be subsampled. The DCT and quantization steps are then performed separately on the three components Y, Cr and Cb. On our proposed approach, encryption can be only applied on the luminance component Y or both on the luminance and the two chrominance

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2019.2894520, IEEE Transactions on Circuits and Systems for Video Technology

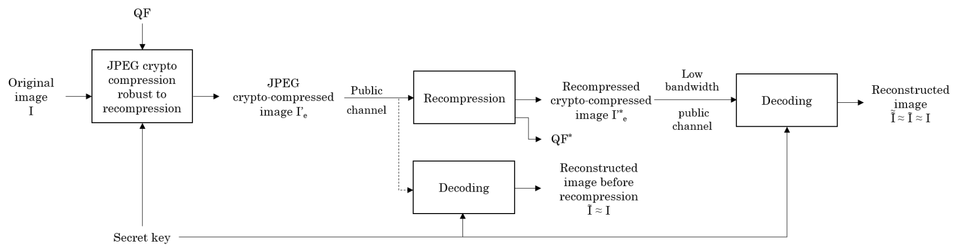


Fig. 2: A global overview of the proposed recompression method of JPEG crypto-compressed images.

components Cr and Cb. This is justified by the fact that the Y component carries the most significant information. In order to preserve the compression rate, we only encrypt the non-zero quantized coefficients [5]. A JPEG crypto-compressed image is then obtained and we are interested in recompressing this encrypted image, without knowing the secret key nor the original image content. The overview of the method is presented in Fig. 2.

Assume that Alice uses a crypto-compression system in order to protect her image against malicious use, and she wants to use format-compliant recompression. In this case, this does not require a specific viewer and it can allow partial, blurred or low resolution visualization in case of selective encryption. Alice does not know the state of the bandwidth, first she can crypto-compress her image in order to let the network provider recompress the image without knowing the secret key. Through the network, the image can then be recompressed to a lower quality factor $\text{QF}^* < \text{QF}$. In addition, with the aim that it can be completed directly in the encrypted domain (*i.e.* without knowing the secret key), our recompression method is performed into the JPEG bitstream. The main idea for the recompression consists of removing the last bit of the amplitude of each non-zero coefficient code, which corresponds to a division by two in base-10. During the decoding phase, it is thus necessary to adapt the quantization table by multiplying each coefficient by two. Moreover, decryption is possible because the removal part in the encrypted sequence can be localized and then, the pseudo-random binary sequence can be resynchronized.

B. *Crypto-compression*

In this paper, for the JPEG crypto-compression step, based on the method described in [5], we propose a new crypto-compression approach in order to make it possible to apply one or multiple recompressions after the crypto-compression. Indeed, if we try to do a recompression of a crypto-compressed image with the method proposed in [5] without any adaptation, then, during the decoding step, there is a desynchronization with the pseudo-random sequence used for decryption. Consequently, the image content in clear can absolutely not be recovered. In order to solve this problem, in our proposed crypto-compression method, we have added a sorting step during the encryption step, in order to make recompression possible after encryption. With this method, the size of the JPEG crypto-compressed image is preserved compared with a standard JPEG compression. An overview of the crypto-compression method is presented in Fig. 3. Until the quantization, the method follows the standard JPEG compression steps. After the quantization, the encryption is completed during the JPEG Huffman coding step. In order ensure minimum requirements of confidentiality, encryption is inevitably applied on the Y component. The two chrominance components can be encrypted, but as illustrated in Fig. 3, they can also remain in clear because they do not carry important visible information. For each MCU, all the $F'(u,v)$, which are non-zero quantized coefficients, are used for encryption. The DC coefficient $F'(0,0)$ consists of a pair $(H_{F'(0,0)}, A_{F'(0,0)})$. The amplitude parameter $A_{F'(0,0)}$ is a code for the prediction error, and the head parameter $H_{F'(0,0)}$ is a simple scalar corresponding to the size of this amplitude. Moreover, all other AC coefficients $F'(u,v)$, such as $(u,v) \neq (0,0)$, are made up of a pair $(H_{F'(u,v)}, A_{F'(u,v)})$, where $A_{F'(u,v)}$ encodes the amplitude. Otherwise, the head $H_{F'(u,v)}$ is composed of the run-length computed previously, and of the amplitude size parameter. Therefore, according to their amplitude size, all non-zero $F'(u,v)$ are sorted to be encrypted, from the largest to the smallest ones with amplitudes equal to 1. This sorting is very important to be able to decode the recompressed JPEG crypto-compressed image without error, as explained in Section III-D.

Indeed, as explained in Section III-C, during the recompression step, every non-zero $F'(u,v)$ coefficients are divided by two. With this proposed reordering, during the decoding, we are still able to resynchronize the pseudo-random generator, even with a $F'(u,v)$ with an amplitude which is encoded on only one bit and which is quantized to zero after a recompression. As we cannot differentiate these coefficients with those that were already null before recompression, without this sorting, there is a desynchronization with the pseudo-random binary sequence. Therefore, in this case the image content in clear cannot be recovered, which is actually the case with the use of the crypto-compression method described in [5].

After the selection and the reordering of the non-zero $F'(u,v)$, a secret key is used to generate a different seed for each MCU. This seed is taken as input of a pseudo-random generator to obtain a pseudo-random binary sequence, according to the size information of each selected coefficient $F'(u,v)$. Indeed, this size is used to determine the required amount of bits to perform the encryption. This sequence is then used to encrypt the amplitude part of the coefficients $F'(u,v)$. The value of the encrypted coefficient $F'_e(u,v)$ is:

$$\begin{aligned} F'_e(u,v) &= E\left(F'(u,v),\ size(F'(u,v))\right), \\ &= E\left(\left\{H_{F'(u,v)}, A_{F'(u,v)}\right\},\ size(F'(u,v))\right), \\ &= \left\{H_{F'_e(u,v)}, A_{F'_e(u,v)}\right\}, \end{aligned} \quad (6)$$

where $H_{F'_e(u,v)} = H_{F'(u,v)}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2019.2894520, IEEE Transactions on Circuits and Systems for Video Technology

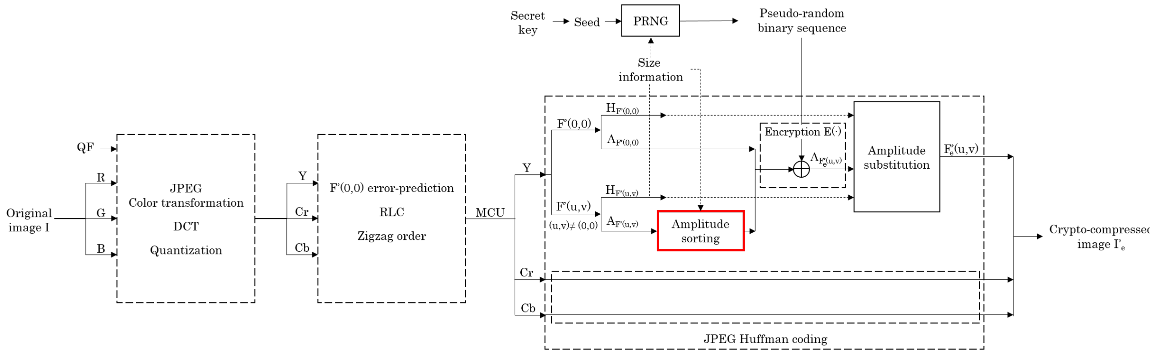


Fig. 3: Overview of the JPEG crypto-compression method which is robust to recompression.

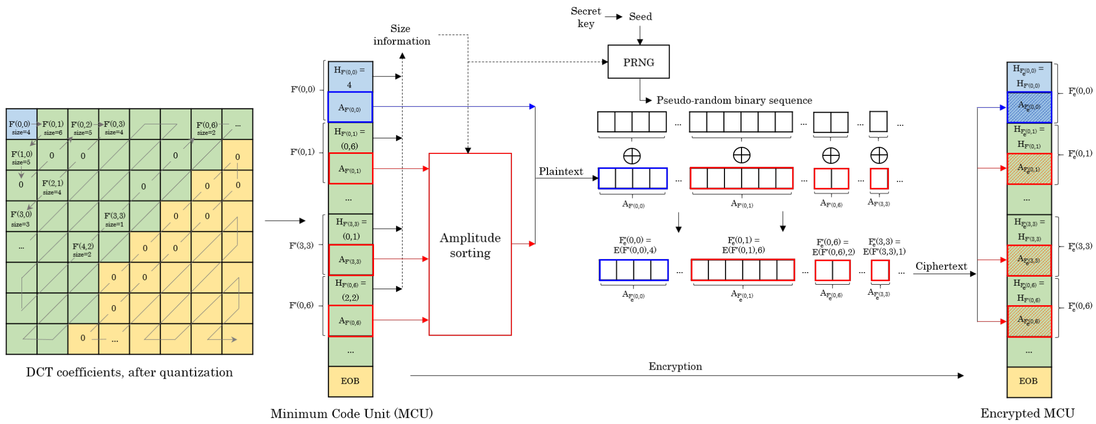


Fig. 4: Example: crypto-compression of one block of quantized DCT coefficients.

As illustrated in Fig. 3, the encryption function $E(\cdot)$ corresponds to a binary XOR-operation between the amplitude value of the clear coefficient $F'(u,v)$ with the corresponding part of the generated pseudo-random binary sequence, according to its size and its position in the MCU. The amplitude values $A_{F'(u,v)}$ of the original bitstream are substituted by the encrypted values $A_{F'_e(u,v)}$ to obtain the encrypted MCU. In this sequence, the encrypted version of the coefficient $F'(u,v)$, coded by a pair $\{H_{F'(u,v)}, A_{F'(u,v)}\}$, is $F'_e(u,v)$, coded by a pair $\{H_{F'_e(u,v)}, A_{F'_e(u,v)}\}$, where the head information remains the same. Note that the clear and the encrypted coefficients have exactly the same number of bits, since we apply only a binary XOR-operation directly on the bits.

In Fig. 4, an example of an application of the proposed crypto-compression method is illustrated for one block. After the quantization step, the quantized DCT coefficients are stored in zigzag order in the MCU. Then, the DC coefficient and all AC coefficients, which are non-zero coefficients, are considered for encryption (for example $F'(0,1)$, $F'(3,3)$ and $F'(0,6)$). These coefficients are sorted as a function of their size, before being considered as the to-be-encrypted bitstream, which starts with the amplitude of the prediction error of $F'(0,0)$. For each MCU, a secret key is used to generate a different seed. This seed is taken as input of a pseudo-random generator to obtain a pseudo-random binary sequence. The amplitude part of each coefficient is then encrypted, with respect to size information. The encrypted amplitude values $A_{F'_e(0,0)}, A_{F'_e(0,1)}, A_{F'_e(3,3)}$ and $A_{F'_e(0,6)}$ are then substituted to the amplitudes $A_{F'(0,0)}, A_{F'(0,1)}, A_{F'(3,3)}$ and $A_{F'(0,6)}$ respectively. The encrypted MCU has exactly the same size as the original one and it can be decoded by a standard viewer, but with a content (the amplitude of the non-zero quantized coefficients $F'(u,v)$) which is encrypted.

C. Recompression of crypto-compressed image

The global scheme of the recompression stage is presented in Fig. 5. As shown, the recompression is applied directly to the encrypted JPEG bitstream for each component. Each MCU, after the encryption phase, is composed of coefficients encoded by pairs head/amplitude $\{H_{F'_e(u,v)}, A_{F'_e(u,v)}\}$ in the MCU. The first step of recompression consists of removing the least significant bit from the amplitude binary representation of each non-zero quantized DCT coefficient. The compressed coefficients $F'^{*}_e(u,v)$ are computed by removing the least significant bit of each coefficient $F'_e(u,v)$:

$$F'^{*}_e(u,v) = \begin{cases} \left\lfloor \frac{F'_e(u,v)}{2} \right\rfloor, & \text{if } |F'_e(u,v)| > 1, \\ 0, & \text{if } |F'_e(u,v)| = 1. \end{cases} \quad (7)$$

Removing the last bit of each coefficient implies reducing the binary size by one for all initially non-zero coefficients. It is also necessary to adjust the size value of the head part of these coefficients, according to the new amplitude value $(size - 1)$. Moreover, when a $F'_e(u,v)$ coefficient has a size of 1 bit before recompression, after the process, its compressed version $F'^{*}_e(u,v)$ is a zero coefficient. Consequently, it is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2019.2894520, IEEE Transactions on Circuits and Systems for Video Technology

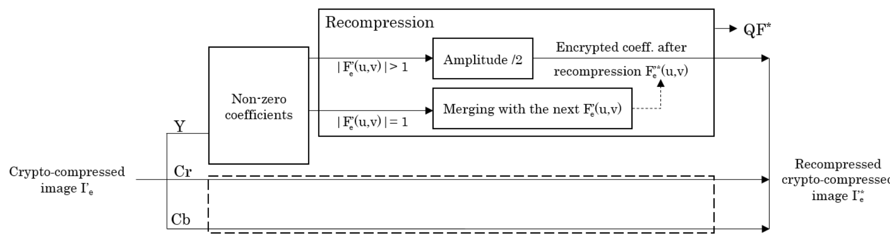


Fig. 5: Recompression of the encrypted JPEG bitstream.

coded in the run-length of the code of the next $F_e'^{*}(u,v)$ coefficient, which is necessarily non-null by construction. Then, the corresponding run-length value of its head part $H_{F_e'(u,v)}$ is adapted depending on the number of previous zeros. Therefore, the new run-length corresponds to the run-length of the current coefficient plus that of the previous coefficient, plus one:

$$H_{F_e'^{*}(u,v)} = (new\ run\text{-}length,\ size - 1). \tag{8}$$

Finally, in each MCU, the recompressed quantized DCT coefficients are encoded by $\{H_{F_e'^{*}(u,v)}, A_{F_e'^{*}(u,v)}\}$. The coefficients $q_{\mathrm{QF}^*}(u,v)$ of the quantization table Q_{QF^*} are:

$$q_{\mathrm{QF}^*}(u,v) = \begin{cases} 2 \times q_{\mathrm{QF}}(u,v), & \text{if } 2 \times q_{\mathrm{QF}}(u,v) \le 255, \\ 255, & \text{otherwise.} \end{cases} \tag{9}$$

In JPEG standard, values of quantization tables are bounded, thus if $q_{\mathrm{QF}^*}(u,v) > 255$, then $q_{\mathrm{QF}^*}(u,v) = 255$, to be fully JPEG format-compliant. Consequently, due to the truncation, image quality can be altered in case of overflow. As the new quantization table Q_{QF^*} is derived from the table Q_{QF} used during the first JPEG compression, the second compression is not obtained according to a predefined standard quality factor. Consequently, it is not possible to choose the desired quality of the resulting compressed image after decryption. This depends directly from the quality factor chosen for the first compression. The problem is therefore to estimate the quality factor after recompression QF^* using the quantization table Q_{QF^*}. In order to give an approximation of this quality factor, we propose to invert Eq. (2) and to compute the value for each coefficient and give an average value. We have two possible equations:

$$\mathrm{EQF}^*_{\le 50} = \left\lfloor \frac{1}{64} \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{q_{50}(u,v) \times 5000}{q_{\mathrm{QF}^*}(u,v) \times 100 - 50} \right\rfloor, \tag{10}$$

$$\mathrm{EQF}^*_{> 50} = \left\lfloor \frac{1}{64} \sum_{u=0}^{7} \sum_{v=0}^{7} 100 - \frac{q_{\mathrm{QF}^*}(u,v) \times 50 - 25}{q_{50}(u,v)} \right\rfloor. \tag{11}$$

Therefore, the estimated QF^*, denoted EQF^*, is given by:

$$\mathrm{EQF}^* = \begin{cases} \mathrm{EQF}^*_{\le 50}, & \text{if } \mathrm{EQF}^*_{\le 50} \le 50, \\ \mathrm{EQF}^*_{> 50}, & \text{otherwise.} \end{cases} \tag{12}$$

This inversion method works if the Eq. (3) is not considered. The range value limitation implies that values which are not included in the interval $[1, 255]$ are lost. Extreme cases are then defined by two quantization tables Q_{QF^-} and Q_{QF^+}, where all coefficients $q_{\mathrm{QF}^-}(u,v)$ and $q_{\mathrm{QF}^+}(u,v)$, $0 \le u, v < 8$ are equal to 1 and 255 respectively. Using Eq. (12), we have $Q_{\mathrm{QF}^-} = 11$ and $Q_{\mathrm{QF}^+} = 99$ and then, $\mathrm{EQF}^* \in [11, 99]$, although $\mathrm{QF} \in [1, 100]$.

However, the main advantage of the proposed method is that it is very easy to localize the removed bits, due to the proposed recompression scheme. In this way, synchronization with the generated pseudo-random binary sequence is still possible and then, the decryption can occur without any error. Indeed, since the non-zero coefficients have been sorted as a function of their amplitudes, after the recompression, the coefficients having a value equal to zero do not desynchronize the bitstream of a MCU since they are at the end of the sequence.

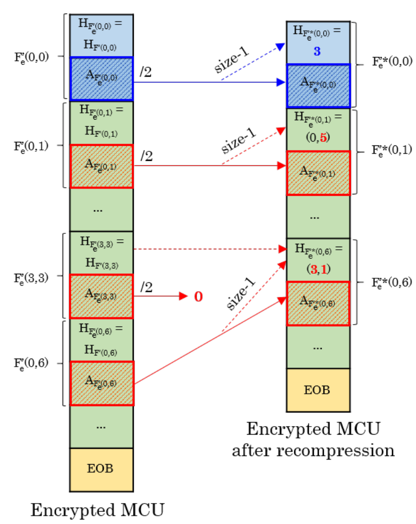


Fig. 6: Our proposed recompression method applied to the example used in Fig. 4.

The recompression step applied to the example in Fig. 4, is presented in Fig. 6. First, the encrypted coefficients $F_e'(0,0)$, $F_e'(0,1)$ and $F_e'(0,6)$ are divided by two. Their size is therefore decreased by one and their associated head values $H_{F_e'(0,0)}$, $H_{F_e'(0,1)}$ and $H_{F_e'(0,6)}$ are modified as a consequence. Note that the amplitude part of the encrypted coefficients with an amplitude equal to 1, like $F_e'(3,3)$, is also divided by two. When a size is one bit before recompression, this coefficient becomes zero coefficient after recompression. Consequently, $F_e'(3,3)$, after recompression, is set to zero and is then included in the entropy coding of the next encrypted coefficient $F_e'^{*}(0,6)$: the run-length of its head part is changed as a function of the run-length of $F_e'(3,3)$. The recompressed crypto-compressed image is still format-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2019.2894520, IEEE Transactions on Circuits and Systems for Video Technology

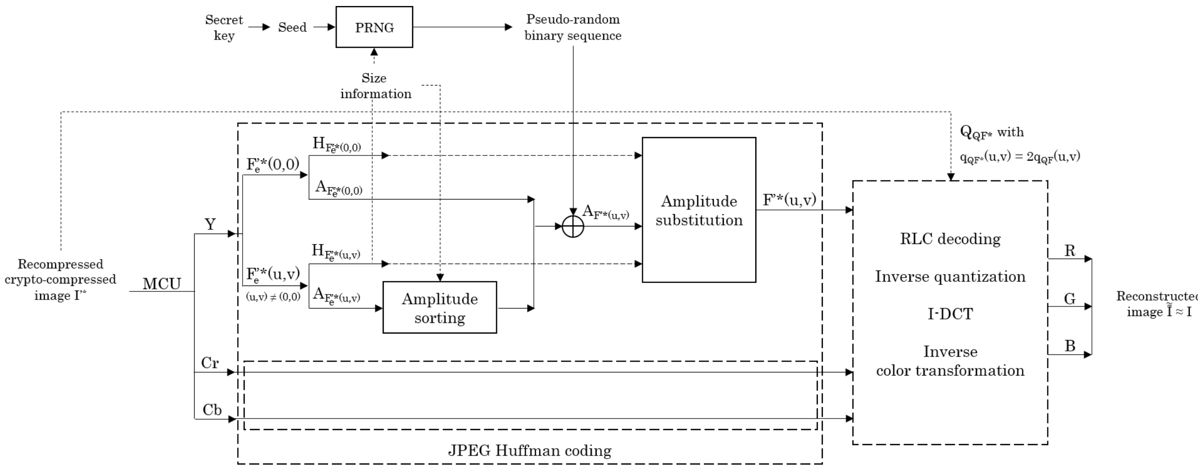


Fig. 7: Overview of the decoding phase.

compliant. Furthermore, the decryption stage can be performed with the secret key used during the encryption phase, but only if the information that the image is recompressed is known. For example, a flag informing how many times the crypto-compressed image has been recompressed can be added in the JFIF comment part.

D. Decoding phase

As presented in Fig. 7, the decoding phase includes four main steps: the decryption of the encrypted JPEG bitstream during the Huffman decoding stage, the inverse quantization operation, the I-DCT transformation and the inverse color transformation.

Decryption can be done by anyone that has the same secret key used during the encryption. Thanks to the JFIF comment part, the user who performs the decryption knows if the encrypted image has been recompressed or not. He first generates the pseudo-random binary sequence, using the secret key as a seed of the pseudo-random generator. After that, he knows that he has to shift the encrypted sequence during the decryption of each coefficient. In fact, as each coefficient has been divided by two during the recompression phase, the last bit was removed and it is necessary to take into account this information, by not considering the last bit of each part of the pseudo-random binary sequence related to one coefficient. The decryption function $D(\cdot)$, similarly to the encryption one, takes two arguments as input: the encrypted coefficient $F_e'^*(u,v)$ and its size according to the head part, which corresponds to the size of the non-encrypted coefficient minus one. It consists of a binary XOR between the amplitude of the encrypted coefficient and the related part of the encrypted sequence:

$$
\begin{aligned}
F'^*(u,v) &= D\left(F_e'^*(u,v),\ size(F_e'^*(u,v))\right), \\
&= D\left(\left\{H_{F_e'^*(u,v)}, A_{F_e'^*(u,v)}\right\},\ size(F'(u,v))-1\right), \\
&= \left\{H_{F'^*(u,v)}, A_{F'^*(u,v)}\right\},
\end{aligned}
\tag{13}
$$

where $H_{F'^*(u,v)} = H_{F_e'^*(u,v)}$.

Note that if we use the proposed recompression method directly on the compressed image I' without encryption, we obtain exactly the same coefficients $F'^*(u,v)$ than after the decryption of the coefficients $F_e'^*(u,v)$. In fact, we have the following relation, because the encryption/decryption method is commutative with the recompression method, as the XOR operation is commutative with the floor function:

$$
\begin{aligned}
F'^*(u,v) &= D\left(\left\lfloor \frac{E(F'(u,v),\ size(F'(u,v)))}{2} \right\rfloor,\ size(F'(u,v))-1\right), \\
&= \left\lfloor \frac{D(E(F'(u,v),\ size(F'(u,v))),\ size(F'(u,v))-1)}{2} \right\rfloor, \\
&= \left\lfloor \frac{F'(u,v)}{2} \right\rfloor .
\end{aligned}
\tag{14}
$$

As explained in Section III-B, since the coefficients becoming zero after recompression are included at the end of the sequence for each block, the decoding phase is error free. Then, even if they have been encrypted previously, there is no mismatch with the bits of the pseudo-random binary sequence and those of the encrypted sequence.

After the decryption step, the Huffman decoding is applied and the quantized DCT coefficients are retrieved. The inverse quantization operation is then performed in order to obtain the dequantized value $\tilde{\tilde{F}}(u,v)$. As explained previously, the decrypted image corresponds to the recompressed compressed image I'^* and its related quantization table Q_{QF^*} is the quantization table Q_{QF} of the compressed image I', whose total coefficients are multiplied by two:

$$
\begin{aligned}
\tilde{\tilde{F}}(u,v) &= F'^*(u,v) \times q_{\text{QF}^*}(u,v) \\
&= \begin{cases} F'^*(u,v) \times 2 \times q_{\text{QF}}(u,v), & \text{if } 2 \times q_{\text{QF}}(u,v) \leq 255, \\ 255, & \text{otherwise.} \end{cases}
\end{aligned}
\tag{15}
$$

Moreover, we can see that the proposed recompression method is better than a naive recompression, as described in Section I. Indeed, each dequantized coefficient $\tilde{\tilde{F}}(u,v)$ of the recompressed image can be defined as a function of the input dequantized coefficient $\tilde{F}(u,v)$, such as:

$$
\begin{aligned}
\tilde{\tilde{F}}(u,v) &= F'^*(u,v) \times q_{\text{QF}^*}(u,v) \\
&= \lfloor \frac{F'(u,v)}{2} \rfloor \times 2 \times q_{\text{QF}}(u,v) \\
&= \begin{cases} \tilde{F}(u,v) & \text{, if } F'(u,v) \text{ is even,} \\ \tilde{F}(u,v) - q_{\text{QF}}(u,v) & \text{, if } F'(u,v) \text{ is odd.} \end{cases}
\end{aligned}
\tag{16}
$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2019.2894520, IEEE Transactions on Circuits and Systems for Video Technology

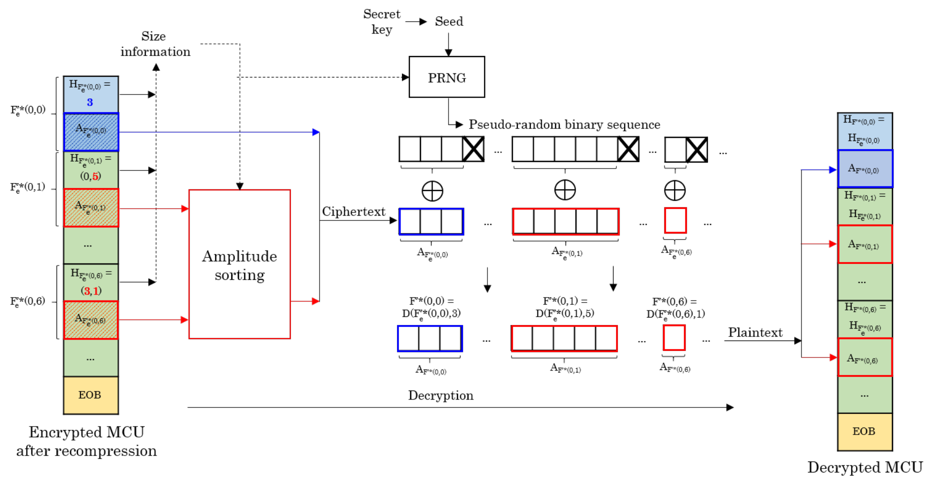


Fig. 8: Decoding method applied to the encrypted MCU after recompression of the example illustrated in Fig. 6.

Therefore, it is shown that in the case of a recompression of an even quantified coefficient, the dequantization of the recompressed coefficient produces the same value as the dequantized coefficient.

Finally, the decompressed RGB image $\tilde{\tilde{I}}$ is obtained by performing the I-DCT transformation, to convert the frequency coefficients into pixels, and then, the inverse color transformation converts the YCrCb image to RGB space. Just like with a standard JPEG compression, as a function of QF, the content of the reconstructed image $\tilde{\tilde{I}}$ is more or less similar to the original image I.

In Fig. 8, we can see the application of the decoding method on the encrypted MCU after recompression of the example illustrated in Fig. 6. First, the secret key – identical to the one involved in the encryption phase – is used to generate the pseudo-random binary sequence. Each encrypted coefficient $F_e^{\prime *}(0,0)$, $F_e^{\prime *}(0,1)$ and $F_e^{\prime *}(0,6)$ is decrypted by XORing its amplitude value with the corresponding part of the generated binary sequence. Indeed, the size parameter gives us the amount of bits to select, and the information that there was a recompression allows us to understand that it is necessary to shift the generated binary sequence between each decryption of one coefficient. The clear coefficients $F^{\prime *}(0,0)$, $F^{\prime *}(0,1)$ and $F^{\prime *}(0,6)$ are obtained by substituting the amplitude values in the MCU. Furthermore, the head part of each coefficient remains the same. At the end of the decryption, we obtain the recompressed compressed image $I^{\prime *}$. To reconstruct the image in the spatial domain $\tilde{\tilde{I}}$, it is necessary to perform the inverse quantization operation, the I-DCT and finally, the inverse color transformation.

E. Application example of the proposed method

To summarize the proposed approach, we present a full example of the proposed method of recompression of JPEG crypto-compressed images. Suppose that $F(0,1) = 164$ is quantized by $q_{80}(0,1) = 4$ for a quality factor of 80%. After the quantization operation, its new value is $F'(0,1) = 41$. Since 41 is in the range $[\![-63,-32]\!] \cup [\![32,63]\!]$, in the Huffman table, its binary amplitude value is encoded by $A_{F'(0,1)} = 101001$. Since, it is preceded by any zero in zigzag order, its corresponding head value is $H_{F'(0,1)} = (0,6)$. Indeed, its run-length is 0 and 6 bits are necessary for the size to encode its amplitude. This pair run-length/size is then encoded by 1111000, according to the standard Huffman table. The binary code of the quantized coefficient $F'(0,1)$ is thus 1111000 101001 (13 bits).

The encryption algorithm $E(\cdot)$ is applied to $F'(0,1)$ to obtain the encrypted coefficient $F_e'(0,1) = E(F'(0,1),6)$. As the size parameter is equal to 6, the corresponding six bit sub-sequence of the pseudo-random binary sequence are selected: for example, 101010. The encrypted value of the amplitude $A_{F_e'(0,1)}$ is computed by XORing this part of the pseudo-random binary sequence and $A_{F'(0,1)}$: $A_{F_e'(0,1)} = 101010 \oplus 101001 = 000011$, being -60 in decimals. The encrypted value of the quantized coefficient $F'(0,1)$ is obtained by substituting this value in the code: $F_e'(0,1) = 1111000\ 000011$. Note that the head is unchanged.

Then, the recompression scheme can be performed. The amplitude value of $F_e'(0,1)$ is divided by two, which corresponds to removing the last bit: $A_{F_e'^*(0,1)} = 00001\cancel{1}$, being -30 in decimal. After that, the head value is adapted in consequence, because the size of the amplitude is now equal to 5. Thus, the head parameter of the recompressed crypto-compressed coefficient is equal to $H_{F_e'^*(0,1)} = (0,5)$, which is encoded by 11010. Finally, $F_e'^*(0,1)$ is encoded on 10 bits: 11010 00001.

During the decoding phase, the decryption function $D(\cdot)$ is applied to $F_e'^*(0,1)$ to compute the clear value $F'^*(0,1) = D(F_e'^*(0,1),5)$. The same part of the pseudo-random binary sequence as those during the encoding phase is used to perform the decryption of the amplitude part, but the last bit is ignored: $A_{F'^*(0,1)} = 10101\cancel{0} \oplus 00001 = 10100$, being 20 in decimal. The decrypted recompressed crypto-compressed coefficient $F'^*(0,1)$ is then encoded by 11010 10100. After the Huffman decoding, the inverse quantization is performed in order to reconstruct the value $\tilde{\tilde{F}}(0,1)$. The quantization table is multiplied by two, so the value $\tilde{\tilde{F}}(0,1) = 20 \times (2 \times q_{80}(0,1)) = 20 \times 8 = 160$. Note that this value is close to the original value $F(0,1) = 164$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2019.2894520, IEEE Transactions on Circuits and Systems for Video Technology

IV. Experimental results

In this section, we present the results we obtained by applying our method of recompression of JPEG crypto-compressed images. Section IV-A gives a full example of application of our method, by using different quality factors for the first JPEG compression. We also show that it is actually possible to recompress several times a JPEG crypto-compressed image. Then, in Section IV-B, we perform an analysis in order to estimate the quality factor of the obtained image after recompression. Finally, in Section IV-C, we discuss level of security and statistical properties of the JPEG crypto-compressed images in order to estimate a visual security of the proposed encryption method. Furthermore, we discuss the parameters to select depending on the required security level and practical applications.

A. A detailed example for the proposed method

We first apply our method on the *Peppers* image (321×481 pixels). Fig. 9 shows the results obtained using $\text{QF} = 75\%$ for the first JPEG compression.

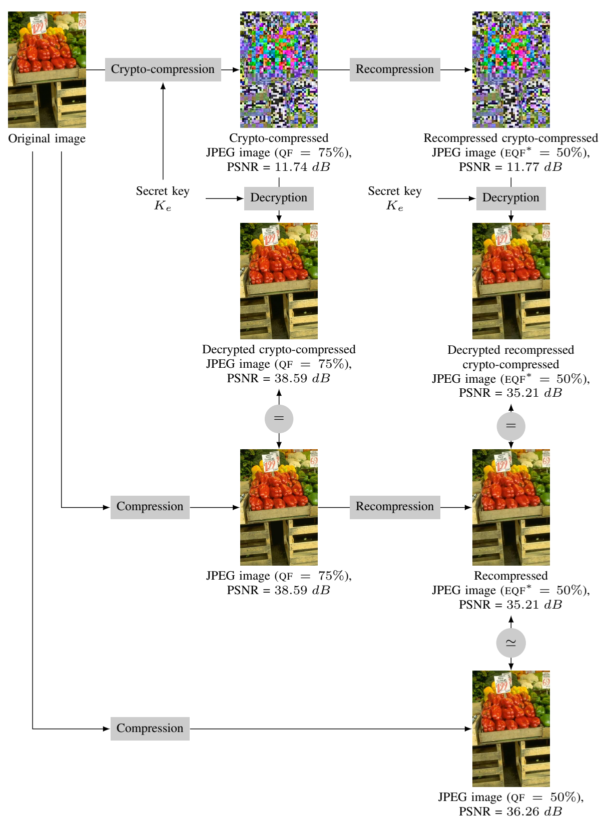


Fig. 9: Full application example of our proposed method: crypto-compression of the *Peppers* image ($\text{QF} = 75\%$, encryption of both AC and DC coefficients of the luminance and two chrominance components) and recompression of the crypto-compressed image ($\text{EQF}^* = 50\%$).

The first step of our method consists of crypto-compressing the original image. In this application example, AC and DC coefficients of the three components (Y, Cr and Cb) are encrypted in order to provide a good visual confidentiality. In fact, we can see that it is really difficult to distinguish details of the original content and we have a very low color PSNR of $11.74\ dB$. After decoding, we can see that the decrypted crypto-compressed JPEG image is very close to the original image, which is indicated by a PSNR equal to $38.59\ dB$. Note that this image is exactly the same as the image obtained with a standard JPEG compression in the clear domain using $\text{QF} = 75\%$. Then, we recompress the obtained crypto-compressed image, directly in the encrypted domain (*i.e.* without decrypting the crypto-compressed image). By analyzing the quantization table, we obtain $\text{EQF}^* = 50\%$. Finally, the recompressed crypto-compressed JPEG image can be perfectly decrypted with the encryption key used during the crypto-compression step. PSNR value is high ($35.21\ dB$), which indicates a strong similarity with the original *Peppers* image.

In order to compare, we also recompress the JPEG image in the clear domain with $\text{QF} = 75\%$ using our recompression method. The obtained recompressed image is identical to the decrypted recompressed crypto-compressed JPEG image. In addition, if the original image is directly compressed (using the standard JPEG compression method) with $\text{QF} = \text{EQF}^*$, we can see that the obtained image is quite close to the decrypted recompressed crypto-compressed JPEG image.

We have completed the same experiments starting with a crypto-compression using $\text{QF} = 50\%$. Then, we obtain $\text{EQF}^* = 25\%$, and we reach the same conclusions as before.

Our method has been applied on 1,338 images from the UCID database [46]. Each image is crypto-compressed, then recompressed and finally decrypted. Fig. 10 presents the compression rate in bit-per-pixel (*bpp*), as a function of the image quality in comparison with the original image (in terms of color PSNR). The plotted values have been obtained by averaging the results from the 1,338 images. For the crypto-compression step, both AC and DC coefficients of the luminance and the two chrominance components have been encrypted and various quality factors QF have been used.

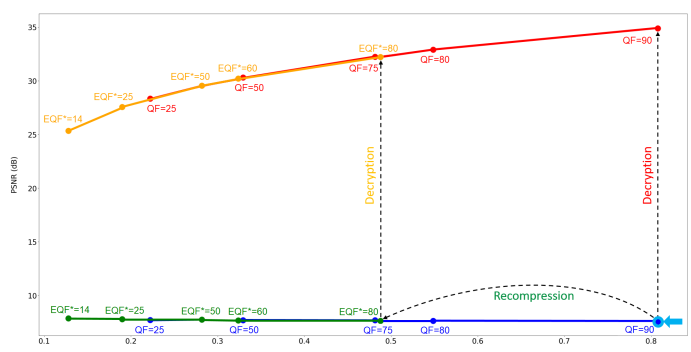


Fig. 10: Average color PSNR for 1,338 images from the UCID database [46] as a function of the average compression rate, in blue the JPEG crypto-compression (various QF, encryption of both AC and DC coefficients of the luminance and the two chrominance components), in green the recompression of the JPEG crypto-compressed images, in red the decryption of the JPEG crypto-compression and in orange the decryption of the recompressed crypto-compressed images.

According to the obtained results in terms of compression rate and image quality, we can see that our method is efficient (PSNR $\simeq 10\ dB$), on being used directly in the encrypted domain. For example, if we perform the first crypto-compression using $\text{QF} = 90\%$, we can see that the compression rate is not very high (approximately $0.9\ bpp$), but the PSNR is close to $35\ dB$, this indicates a strong similarity with the original image content. However, in order to decrease the size of the crypto-compressed image, we can apply the proposed recompression method directly to the crypto-compressed image. EQF^* is also equal to 80% and we achieve this by having a compression rate of approximately $0.53\ bpp$, while maintaining good image quality (PSNR $\simeq 32\ dB$). After recompression, there is still no information about the content of the original image, which is indicated by a very low PSNR value (PSNR $\simeq 10\ dB$). In addition, the PSNR value of the decrypted recompressed crypto-compressed JPEG image is high and remains greater than $30\ dB$ and very close to the direct compressed version. Thus, the proposed method achieves a very good trade-off between the reconstructed image quality and the compression rate, while offering a good level of security because the recompression step occurs directly in the encrypted domain and has no impact on the confidentiality of the original image content.

In Fig. 11, we have applied five times our recompression method on the *Hats* image of 768×512 pixels ($491\ kB$). First, the original image is crypto-compressed by encrypting both AC and DC coefficients of the luminance and the two chrominance components. The initial QF is chosen high ($\text{QF} = 95\%$). If we directly decrypt this crypto-compressed JPEG image, we obtain an image very similar to the original one, as indicated by a PSNR value equal to $47.14\ dB$. We apply then our proposed recompression method on the crypto-compressed JPEG image: all the non-zero coefficients are then divided by two. After this second quantization, some of them become equal to zero and are thus coded in the run-length of the code of the next coefficient. EQF^* is equal to 90% and, if we decrypt the recompressed crypto-compressed JPEG image, the obtained image remains similar to the original one (PSNR = $43.91\ dB$). Note that the recompressed crypto-compressed JPEG image is still suitable for recompression: we can recompress it in order to decrease its size once again. In fact, with our method, it is possible to recompress a crypto-compressed image several times. In this example, we have recompressed the crypto-compressed JPEG image five times. After this series of recompressions, the estimated quality factor is $\text{QF}^* = 17\%$ and the image size is equal to $15.4\ kB$, which corresponds to a compression rate of $0.32\ bpp$. We can remark that it would be possible to obtain this image directly from the initial crypto-compressed image by dividing all coefficients by $2^5 = 32$ and by adapting the comment part JCOM of the JFIF header. Actually, a flag which indicates the number of recompressions is necessary. Thanks to this flag, it is possible to know the number of to-be-shifted bits in the pseudo-random sequence, and thus, the decoding phase is done without error. Moreover, the quality of the associated decrypted recompressed crypto-compressed JPEG image is still high (PSNR = $32.11\ dB$).

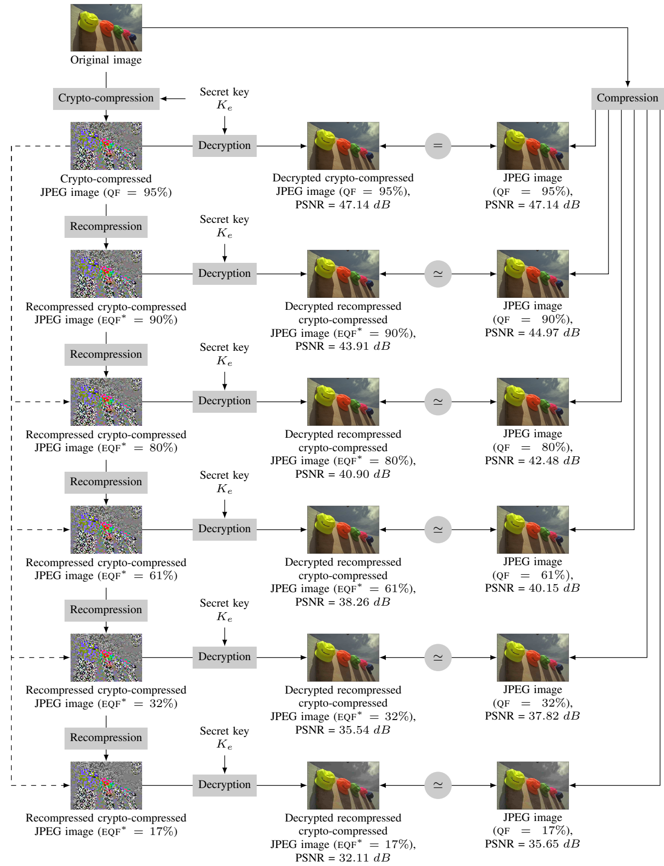


Fig. 11: Five recompressions of the *Hats* image starting with $\text{QF} = 95\%$ for the crypto-compression (encryption of both AC and DC coefficients of the luminance and the two chrominance components).

B. Quality factor analysis

In this section, we discuss the estimation of the quality factor after recompression EQF^*. This estimation is made from the luminance quantization table, because it is more relevant than using the chrominance quantization table. In Eq. (12), we have shown that it is possible to obtain the value of EQF^* by inverting Eq. (2). Due to the range limitation, we have noticed that $\text{EQF}^* \in [11, 99]$. In Table I, we present some QF which can be used for the first crypto-compression and the corresponding values of EQF^*. Note that the chosen values are representative of the interval of possible values. We can see that, for $\text{QF} \geq 90\%$, EQF^* remains high ($\simeq -10\ \%$). For small QF ($\text{QF} \leq 25\%$), values of EQF^* are also close to QF ($\simeq -10\ \%$). For widely used QF (for example, $\text{QF} = 75\%$ and $\text{QF} = 50\%$), we note that EQF^* is much lower than before recompression (from $-25\ \%$ to $-50\ \%$).

QF (%)	100	95	90	75	50	25	15
EQF* (%)	97	90	80	50	25	14	12

TABLE I: Example of QF and their corresponding EQF^* after recompression using our proposed method

In Fig. 12, we illustrate the difference between the quantization table Q_{QF^*} obtained with our recompression method and associated to the estimated quality factor EQF^*, and the quantization table Q_{QF}, such as $\text{QF} = \text{EQF}^*$. As an example, in

Fig. 12.b, we present Q_{75}. This quantization table, generated from Q_{50} (Fig. 12.a) according to Eq. (2), is used during the crypto-compression of an original image. Using our proposed recompression method, Q_{QF^*} is computed by multiplying each coefficient of Q_{QF} by two (Eq. (9)). The obtained table is presented in Fig. 12.c. Moreover, using Eq. (12), EQF^* is equal to 50. By comparing Fig. 12.a and Fig. 12.c, we can see that the two tables are very similar. Indeed, the difference between two coefficients at the same position is either null or equal to one.

	0	1	2	3	4	5	6	7
0	16	11	10	16	24	40	51	61
1	12	12	14	19	26	58	60	55
2	14	13	16	24	40	57	69	56
3	14	17	22	29	51	87	80	62
4	18	22	37	56	68	109	103	77
5	24	35	55	64	81	104	113	92
6	49	64	78	87	103	121	120	101
7	72	92	95	98	112	100	103	99

(a)

	0	1	2	3	4	5	6	7
0	8	6	5	8	12	20	26	31
1	6	6	7	10	13	29	30	28
2	7	7	8	12	20	29	35	28
3	7	9	11	15	26	44	40	31
4	9	11	19	28	34	55	52	39
5	12	18	28	32	41	52	57	46
6	25	32	39	44	52	61	60	51
7	36	46	48	49	56	50	52	50

(b)

	0	1	2	3	4	5	6	7
0	16	12	10	16	24	40	52	62
1	12	12	14	20	26	58	60	56
2	14	14	16	24	40	58	70	56
3	14	18	22	30	52	88	80	62
4	18	22	38	56	68	110	104	78
5	24	36	56	64	82	104	114	92
6	50	64	78	88	104	122	120	102
7	72	92	96	98	112	100	104	100

(c)

Fig. 12: Difference between Q_{50} and Q_{QF^*}, with $\text{EQF}^* = 50\%$: a) Standard quantization table Q_{50}, for $\text{QF} = 50\%$, b) Quantization table Q_{75}, for $\text{QF} = 75\%$, calculated from Q_{50}, c) Quantization table Q_{QF^*} with $\text{EQF}^* = 50\%$, calculated from Q_{75}.

In Fig. 13, in order to deal with this analysis in depth, we evaluate the $L2$-distance between Q_{QF^*} and Q_{QF}, such as $\text{QF} = \text{EQF}^*$ for different values in the interval $[11, 99]$. In other words, we aim to evaluate the relevance of our estimation (EQF^*) from the real value of QF^*. We note that a significant divergence starts for an EQF^* 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 quality factor after recompression from 99% to 34%. Note that for chrominance quantization table, whose coefficients are higher, the divergence is more important.

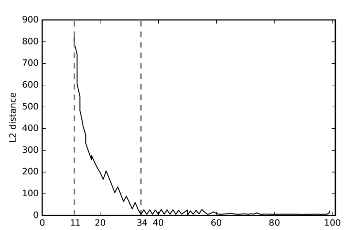


Fig. 13: $L2$-distance between Q_{QF^*} (associated to the estimated quality factor EQF^*), and Q_{QF} calculated from the standard quantization table Q_{50}, such as $\text{QF} = \text{EQF}^*$.

From this experimental result, we note that EQF^* is significant. Therefore, we can expect that the crypto-compressed JPEG image after recompression, associated to EQF^*, would be similar to the image obtained with a direct JPEG crypto-compression with $\text{QF} = \text{EQF}^*$, both in terms of compression rate and image quality, as illustrated Fig. 11.

C. Security analysis discussion

In this part, we propose to discuss the security level of the crypto-compression scheme used in our proposed method of recompression of crypto-compressed JPEG images. As shown in the presentation of the proposed method in Section III and in Fig. 14, different parameters can be used during the crypto-compression of the original image: encryption of AC coefficients or encryption on both AC and DC coefficients, of the luminance component (Y) or on both the luminance and chrominance components (Y, Cr, Cb). These parameters are chosen as a function of the required security level: transparent encryption, sufficient encryption or content confidentiality level [41].

By encrypting only the non-zero AC coefficients of the luminance component (or of the luminance and the two chrominance components), see Fig. 14, we can observe that only a high quality version of the original image is hidden: in this case, we have a transparent encryption. The method follows the requirements created by Van Droogenbroeck in [47] for selective encryption in real-time applications: visual acceptance (part of the information may be visible, but the encrypted image looks noisy), constant bit rate and bitstream compliance. Moreover, sufficient encryption can be achieved by encrypting both AC and DC coefficients of the luminance component only. As illustrated in Fig. 14, in this case, the original content is highly distorted, but color information about the original image content is preserved. However, for the highest security level, it is necessary to hide the image content (content confidentiality level). Therefore, it is required to encrypt both AC and DC coefficients of the luminance and two chrominance components. Using this method, encryption achieves a strong level of security, because only a limited amount of information is available about the original image content from the crypto-compressed image.

In regard to the statistical properties of the encrypted image, we can see that even if the PSNR with the original image is equal to $11.69\ dB$, UACI and NPCR values are not significant (17.61% and 98.14% respectively). Moreover, entropy value is higher than for the original image ($7.61\ bpp > 7.12\ bpp$), but not close to the maximal entropy value of $8\ bpp$. With the χ^2 test, we also observe that the value remains high after encryption (square-root equal to 165.95). In order to enhance the security level by introducing diffusion, it would be possible to apply scrambling in addition to our method – like the full inter-block shuffle (FIBS) [10] or the method of Lian *et al.* [48] for example. In this case, the encryption method should be indistinguishable under chosen plaintext attack (IND-CPA secure). Nevertheless, the size of the encrypted image could increase (but the additional cost remains low). However, even with this improvement, the encryption method cannot be IND-CPA$ secure [1]. Indeed, in addition to being IND-CPA, a crypto-system is IND-CPA$ secure when an adversary cannot make the distinction between an encrypted image and a sequence of random numbers. Actually, this cannot be the case for any selective encryption method which has to be format-compliant, since the JPEG structure must be preserved.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2019.2894520, IEEE Transactions on Circuits and Systems for Video Technology

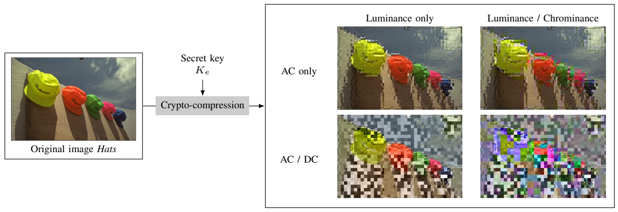
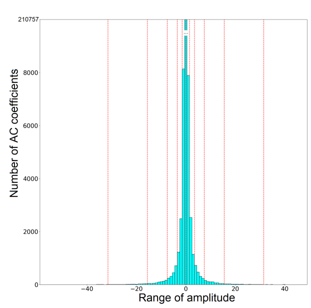
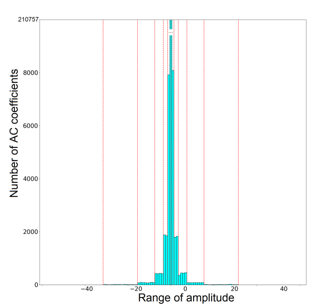


Fig. 14: Crypto-compression of the *Hats* image with QF $= 80\%$ and the possible parameters of our method.

In Fig. 15, we have applied the encryption on the AC and the DC coefficients of the luminance and two chrominance components, by using a quality factor QF $= 90\%$. We present the distribution of the AC coefficients (after quantization) before and after encryption. Firstly, we note that in both cases, the distribution seems to be Laplacian. In fact, there are many coefficients which are equal to zero after JPEG compression. It is also important to notice that the encrypted coefficients have exactly the same size (in terms of bits) than the clear ones. In fact, the size of each to-be-encrypted coefficient is considered as a parameter in order to select the appropriate number of bits in the pseudo-random binary sequence to perform the XOR-operation. In this way, the encryption method allows us to preserve the JPEG structure and the size of the standard JPEG compressed image. From a security point of view, this is a leak, because coefficients in large quantities are encoded with a smaller number of bits (Huffman coding). For example, coefficients equal to 1 or -1 are encoded on only one bit, and there are thus two possible values to encrypt these coefficients. On the positive side, if we consider coefficients which are encoded with the same number of bits, we can see that the encryption process provides uniform distribution (Fig. 15.b). It is thus not possible to exploit them in order to statistically try to reconstruct the clear coefficient values.



(a) Before encryption.



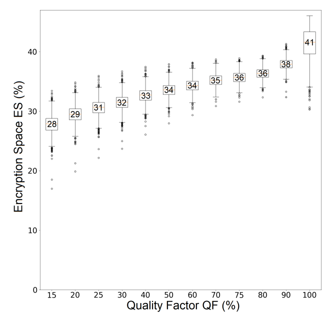
(b) After encryption.

Fig. 15: Distribution of the AC coefficients before and after encryption (QF $= 90\%$, encryption of both AC and DC coefficients of the luminance and two chrominance components).

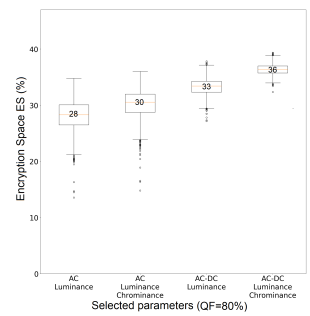
In Fig. 16, we analyzed the encryption space ES as a function of the chosen quality factor for the crypto-compression step, and as a function of the selected parameters (AC, DC, luminance and chrominance):

$$\text{ES} = \frac{\text{number of encrypted bits}}{\text{size of the compressed image (in bits)}}.$$

We can first observe that the higher the quality factor, the larger the ES is. This is explained by the fact that there are more non-zero coefficients with a high QF. Actually, only the non-zero coefficients are encrypted (Fig. 16.a). Then, we can see that the ES varies between 41% for QF $= 100\%$ and 28% for QF $= 15\%$ (on average). The size of the ES is also different as a function of the parameters of our method (see Fig. 16.b). Indeed, with QF $= 80\%$ and by encrypting both AC and DC coefficients, 33% of the image content (on average) is encrypted when the encryption is only performed on the luminance component. Moreover, the ES is higher when the two chrominance components are also encrypted (36% on average). In the case of encrypting only the AC coefficients, on average 28% of the data is encrypted when the encryption is completed only on the luminance component, and 30% when all components are encrypted. However, there is a more important variability depending on the original image content. We can see that a larger amount of information is encrypted if the three components are encrypted, rather than when we encrypt the luminance component only. But the additional encrypted information amount is not important, due to the subsampling of the chrominance components.



(a)



(b)

Fig. 16: ES in $\%$ for our method of recompression of JPEG crypto-compressed images (1,338 images from the UCID database [46]) as a function of: a) QF for the crypto-compression method with encryption of both AC and DC coefficients of the luminance and two chrominance components, b) The selected parameters (with QF $= 80\%$).

V. Conclusion

In this work, we proposed a new method of recompressing crypto-compressed JPEG images, which is efficient in the encrypted domain. From our knowledge, this is one of the first methods allowing recompression directly in the encrypted domain, without knowing the secret key. Recompression step consists mainly in dividing by two each quantized encrypted DCT coefficient. In fact, the least significant bit of the non-zero quantized encrypted coefficients are thus removed, and zero coefficients are then encoded in the run-length of the next non-zero coefficients. For the decoding, the coefficients of the quantization table are adapted in consequence, by multiplying them by two. As shown in the experimental part, this recompression operation achieves a very good trade-off between the compression rate and the image quality. Moreover, unlike standard recompression with JPEG, the recompressed image with EQF* is very similar to the JPEG image obtained with a direct JPEG compression with the equivalent QF. There are no artifacts, such as grainy effects or an important loss

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2019.2894520, IEEE Transactions on Circuits and Systems for Video Technology

of sharpness. In order to make this recompression operation possible in the encrypted domain, the crypto-compression step is adapted. In fact, quantized DCT coefficients are encrypted according to their size, from the largest to the smallest, in order to avoid desynchronization during the decryption phase. Furthermore, in the crypto-compressed image, the main content of the original image is kept secret, as indicated by a PSNR close to 10 dB. Note that, after recompression, visual confidentiality is still preserved, because our recompression method does not introduce security leaks. Therefore, in addition to offering a strong security level and allowing recompression, the used encryption procedure is format-compliant and does not introduce overhead.

In future work, we are interested in investigating other crypto-compression techniques which allow us to apply our proposed recompression method without decryption. In fact, to transform the proposed method into a IND-CPA secure one, it is actually possible to combine it with a scrambling method like, for example, the full inter-block shuffle (FIBS). Moreover, we are also involved in analyzing more precisely the EQF*.

REFERENCES

[1] K. He, C. Bidan, G. L. Guelvouit, and C. Feron, "Robust and secure image encryption schemes during JPEG compression process," *Electronic Imaging*, vol. 2016, no. 11, pp. 1–7, 2016.

[2] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. XVIII–XXXIV, 1992.

[3] A. Unterweger and A. Uhl, "Length-preserving Bit-stream-based JPEG Encryption," in *Proceedings of the on Multimedia and security*. ACM, 2012, pp. 85–90.

[4] 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 (ICIP)*. IEEE, 2012, pp. 261–264.

[5] W. Puech, A. Bors, and J. Rodrigues, "Protection of colour images by selective encryption," in *Advanced Color Image Processing and Analysis*. Springer, 2013, pp. 397–421.

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

[7] S. Ong, K. Wong, X. Qi, and K. Tanaka, "Beyond format-compliant encryption for JPEG image," *Signal Processing: Image Communication*, vol. 31, pp. 47–60, 2015.

[8] A. Said, "Measuring the strength of partial encryption schemes," in *IEEE International Conference on Image Processing (ICIP)*, vol. 2. IEEE, 2005, pp. 1126–1129.

[9] M. Pinto, W. Puech, and G. Subsol, "Protection of JPEG compressed e-comics by selective encryption," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2013, pp. 4588–4592.

[10] W. Li and Y. Yuan, "A leak and its remedy in JPEG image encryption," *International Journal of Computer Mathematics*, vol. 84, no. 9, pp. 1367–1378, 2007.

[11] K. Minemura, K. Wong, R. C.-W. Phan, and K. Tanaka, "A novel sketch attack for H. 264/AVC format-compliant encrypted video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 11, pp. 2309–2321, 2017.

[12] D. Schonberg, S. Draper, and K. Ramchandran, "On compression of encrypted images," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2006, pp. 269–272.

[13] R. Lazzeretti and M. Barni, "Lossless compression of encrypted grey-level and color images," in *IEEE European Signal Processing Conference (EUSIPCO)*. IEEE, 2008, pp. 1–5.

[14] 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 (VCIP), 2013*. IEEE, 2013, pp. 1–6.

[15] S. Chan, "Recompression of still images," *University of Kent, Canterbury, U.K.*, vol. Tech. Rep. 2-92, 1992.

[16] 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*, vol. 12, no. 7, pp. 843–849, 2003.

[17] M. Carnein, P. Schöttle, and R. Böhme, "Telltale watermarks for counting JPEG compressions," *Electronic Imaging*, vol. 2016, no. 8, pp. 1–10, 2016.

[18] A. Lewis and M. G. Kuhn, "Exact JPEG recompression." *Visual Information Processing and Communication*, vol. 7543, 2010.

[19] E. Hamilton, "JPEG file interchange format," 2004.

[20] W. Trappe and L. C. Washington, *Introduction to cryptography with coding theory*. Pearson Education India, 2006.

[21] K. Usman, H. Juzoji, I. Nakajima, S. Soegidjoko, M. Ramdhani, T. Hori, and S. Igi, "Medical image encryption based on pixel arrangement and random permutation for transmission security," in *International Conference on e-Health Networking, Application and Services*. IEEE, 2007, pp. 244–247.

[22] P. Premaratne and M. Premaratne, "Key-based scrambling for secure image communication." in *International Conference on Intelligent Computing*. Springer, 2012.

[23] C. V. Wright, W.-C. Feng, and F. Liu, "Thumbnail-preserving encryption for JPEG," in *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2015, pp. 141–146.

[24] G. Chen, Y. Mao, and C. K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps," *Chaos, Solitons & Fractals*, vol. 21, no. 3, pp. 749–761, 2004.

[25] Y. Mao, G. Chen, and S. Lian, "A novel fast image encryption scheme based on 3D chaotic baker maps," *International Journal of Bifurcation and chaos*, vol. 14, no. 10, pp. 3613–3624, 2004.

[26] Z.-H. Guan, F. Huang, and W. Guan, "Chaos-based image encryption algorithm," *Physics Letters A*, vol. 346, no. 1, pp. 153–157, 2005.

[27] T. Xiang, K.-W. Wong, and X. Liao, "Selective image encryption using a spatiotemporal chaotic system," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 17, no. 2, p. 023115, 2007.

[28] Z.-L. Zhu, W. Zhang, K.-W. Wong, and H. Yu, "A chaos-based symmetric image encryption scheme using a bit-level permutation," *Information Sciences*, vol. 181, no. 6, pp. 1171–1186, 2011.

[29] G. Chen and T. Ueta, "Yet another chaotic attractor," *International Journal of Bifurcation and chaos*, vol. 9, no. 07, pp. 1465–1466, 1999.

[30] C. Shi and B. Bhargava, "A fast MPEG video encryption algorithm," in *Proceedings of the Sixth ACM International Conference on Multimedia*, ser. MULTIMEDIA '98. New York, NY, USA: ACM, 1998, pp. 81–88.

[31] M. Van Droogenbroeck and R. Benedett, "Techniques for a selective encryption of uncompressed and compressed images," in *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS) 2002, Ghent, Belgium*, 2002, pp. 90–97.

[32] W. Puech and J. M. Rodrigues, "Crypto-compression of medical images by selective encryption of DCT," in *IEEE European Signal Processing Conference (EUSIPCO)*. IEEE, 2005, pp. 1–4.

[33] Z. Shahid, M. Chaumont, and W. Puech, "Fast protection of H. 264/AVC by selective encryption of CAVLC and CABAC for I and P frames," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 5, pp. 565–576, 2011.

[34] W. Hamidouche, M. Farajallah, N. Sidaty, S. El Assad, and O. Deforges, "Real-time selective video encryption based on the chaos system in scalable HEVC extension," *Signal Processing: Image Communication*, vol. 58, pp. 73–86, 2017.

[35] F. Dufaux and T. Ebrahimi, "Scrambling for privacy protection in video surveillance systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 8, pp. 1168–1174, 2008.

[36] K. Kurihara, S. Imaizumi, S. Shiota, and H. Kiya, "An encryption-then-compression system for lossless image compression standards," *IEICE TRANSACTIONS on Information and Systems*, vol. 100, no. 1, pp. 52–56, 2017.

[37] T. Chuman, K. Kurihara, and H. Kiya, "On the security of block scrambling-based ETC systems against jigsaw puzzle solver attacks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2157–2161.

[38] K. Kurihara, S. Shiota, and H. Kiya, "An encryption-then-compression system for JPEG standard," in *Picture Coding Symposium (PCS), 2015*. IEEE, 2015, pp. 119–123.

[39] O. Watanabe, A. Uchida, T. Fukuhara, and H. Kiya, "An encryption-then-compression system for JPEG 2000 standard," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 1226–1230.

[40] J. Zhou, X. Liu, O. C. Au, and Y. Y. Tang, "Designing an efficient image encryption-then-compression system via prediction error clustering and random permutation," *IEEE Transactions of Information Forensics and Security*, vol. 9, no. 1, pp. 39–50, 2014.

[41] D. Engel, T. Stütz, and A. Uhl, "A survey on JPEG2000 encryption," *Multimedia systems*, vol. 15, no. 4, pp. 243–270, 2009.

[42] F. Dufaux, S. Wee, J. Apostolopoulos, and T. Ebrahimi, "JPSEC for Secure Imaging in JPEG 2000," in *SPIE Proc. Applications of Digital Image Processing XXVII*, no. LTS-CONF-2004-041. SPIE, 2004.

[43] R. Norcen and A. Uhl, "Encryption of wavelet-coded imagery using random permutations," in *IEEE International Conference on Image Processing (ICIP)*, vol. 5. IEEE, 2004, pp. 3431–3434.

[44] S. Lian, J. Sun, and Z. Wang, "Perceptual cryptography on JPEG2000 compressed images or videos," in *International Conference on Computer and Information Technology*. IEEE, 2004, pp. 78–83.

[45] M. Grangetto, E. Magli, and G. Olmo, "Multimedia selective encryption by means of randomized arithmetic coding," *IEEE Transactions on Multimedia*, vol. 8, no. 5, pp. 905–917, 2006.

[46] G. Schaefer and M. Stich, "UCID – an uncompressed color image database," in *Proceedings of SPIE in Storage and Retrieval Methods Applications for Multimedia*, vol. 5307, 2004, pp. 472–480.

[47] M. Van Droogenbroeck, "Partial encryption of images for real-time applications," in *Fourth IEEE Signal Processing Symposium*, 2004, pp. 11–15.

[48] S. Lian, J. Sun, and Z. Wang, "A novel image encryption scheme based-on JPEG encoding," in *IEEE International Conference on Information Visualisation*. IEEE, 2004, pp. 217–220.

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. He was teaching assistant at the University of Montpellier, France in 2016. He did one year Post-Doctoral at the IMT Lille Douai. Currently, he is a Post-Doctoral Researcher at the University of Montpellier. His research interests lie on visual data processing and multimedia security.

Pauline Puteaux received the M.S. degree in Computer Science and Applied Mathematics, with specialization in Cybersecurity, from the University of Grenoble, France, in 2017. She is currently pursuing a Ph.D. degree with the Laboratory of Informatics, Robotics and Microelectronics of Montpellier, France. Her work has focused on multimedia security, and in particular, image analysis and processing in the encrypted domain.

William Puech received the diploma of Electrical Engineering from the Univ. Montpellier, France (1991) and a Ph.D. Degree in Signal-Image-Speech from the Polytechnic National Institute of Grenoble, France (1997) with research activities in image processing and computer vision. He served as a Visiting Research Associate to the University of Thessaloniki, Greece. From 1997 to 2008, he has been an Associate Professor at the Univ. Montpellier, France. Since 2009, he is full Professor in image processing at the Univ. Montpellier, France.