DOI: 10.2352/ISSN.2470-1173.2016.11.IMAWM-461

Robust and Secure Image Encryption Schemes During JPEG Compression Process

Kun He; B-Com & CentraleSupélec; Rennes, France
Christophe Bidan; B-Com & CentraleSupélec; Rennes, France
Gaëtan Le Guelvouit; B-Com; Rennes, France
Cyrielle Feron; B-Com; Rennes, France

Abstract

While public image-sharing platforms enable users to share images easily, security of published images becomes more important. An effective method to protect a published image is encrypting it before its upload. However, the encrypted result has to be viewed as a correct digital data by the sharing platform. Normally, the digital data should be an image (and often a JPEG image). Therefore, an encryption method which can preserve the image format after encryption is needed. And the main objective of decryption is to reconstruct an image while preserving image quality.

In this paper, we propose three JPEG image encryption schemes and compare them. All these three schemes use the same encryption algorithm but are integrated into different steps of the JPEG compression process. The first one encrypts image before DCT in spatial domain. The second one encrypts image after DCT in frequency domain. The third one encrypts image after quantization. These three schemes undergo JPEG processing and obtain JPEG image as the encrypted result. Further, we get the performances of these three schemes through comparing the runtime, Peak Signal to Noise Ratio (PSNR) and Universal Image Quality Index (UIQI), and conclude that which one is closest to our needs.

Introduction

With the development of digital imaging applications and image-sharing platforms, people are increasingly concerned about the security and privacy of their personal information. But why do we still require new encryption technologies to protect image? It can be explained as: if an image is encrypted using a traditional cipher like AES, thanks to their security, the confidentiality of the image can be ensured. However, the encrypted result is not an image, but a binary file. Considering that digital imaging applications and image-sharing platforms do not accept any file format other than image, the encryption algorithm should preserve the image format after encryption. Obviously, these traditional cipher algorithms are not suitable here.

In this paper, we propose three JPEG image encryption schemes. They use the same encryption algorithm but are integrated into different steps of JPEG compression process and all undergo JPEG processing. The encrypted results are JPEG images, which can be accepted by any image-sharing platform. And the decryption algorithm can reconstruct an image whose quality does not change too much as compared with the original image. Further, we get the performances of these three schemes through comparing their different aspects, and conclude which one is the best.

The rest of the paper is organized as follows. The second section enumerates some related works. The third section introduces basic knowledge of JPEG compression. The fourth section describes the encryption and decryption algorithms. The fifth section proves the security of our algorithm. The sixth section details the three schemes and presents some experimental results to compare them. The last section summarizes this paper.

Related Work

Various data encryption algorithms have been proposed and widely used, such as AES, DES, RSA, etc. Most of them are used for text or binary data and provide very good performances. It is also possible to use them in image encryption scheme. Dang *et al.* [2] choose DES to encrypt the compressed image data. And an image encryption system based on Hill cipher is proposed in [3]. However, the results of encryption are not displayed as an image and cannot be accepted by image-sharing platforms.

So many early researches to protect image was based on scrambling encryption. In spatial domain, the scrambling can be applied to the bits of pixels [4], or directly applied to the pixel of the plaintext image [5]. In frequency domain, most researches choose to permute all or a part of the coefficients [6], but also sometimes only permute the sign of coefficients [7]. Although scrambling encryption can make sure that the encrypted result is a non-intelligible image, which prevents human or even computer vision system from understanding the real content, this technique cannot be seen as a real cipher since it is not secure [8].

There are also many methods that use real cipher to encrypt images and can supply an image as a result. For example, [9] relies on orthogonal matrices to encrypt DCT blocks in frequency domain. However, this method needs a great amount of iterations and calculations. From its experiment, the execution time required to encrypt a grayscale image of 256×256 is about $600 \sim 800$ ms. That means it is not an efficient method. Backes *et al.* [10] propose an algorithm which encrypts plaintext image first using AES and then embeds the encrypted data into a container JPEG image (that is a technique called steganography). But for an 8-bit container JPEG image, only 2 bits are available to embed data. In other word, only 25% encrypted data can be embedded. Therefore, there is a great limitation of the size of container image and the quantity of upload image. In [11], Lian *et al.* assert that the most feasible way is to combine image encryption algorithm with JPEG encoding. They propose a method which permutes the DCT blocks first and then encrypts one sign bit of each coefficient with a chaotic sequence. But by observing the encrypted image, we can clearly understand the color composition of plaintext im-

DOI: 10.2352/ISSN.2470-1173.2016.11.IMAWM-461

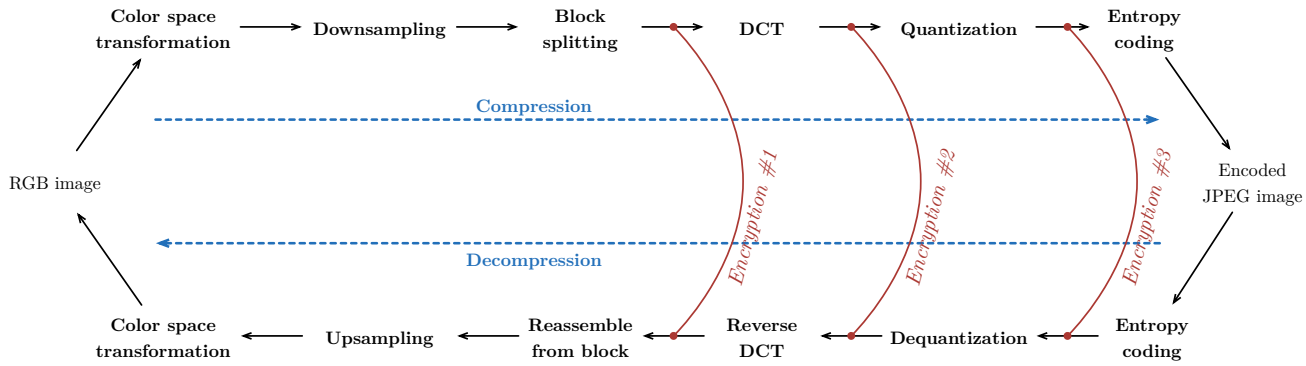


Figure 1. *JPEG compression and decompression processes.*

age. If we use a darker image and a brighter image as examples, it is easy to know which encrypted image corresponds to which original image. So this method cannot ensure a high level of security, especially, it is not IND-CPA secure [12]. In this paper, we propose an image encryption algorithm which is integrated into three different steps of JPEG compression process and obtains a JPEG image as the encrypted result.

JPEG Compression Process

JPEG [1] is one of the widely used standards for storing and compressing image. Digital cameras and other image capture devices use JPEG standard to store image. Images on the World Wide Web are compressed by using JPEG standard as well. Fig. 1 illustrates the complete JPEG compression and decompression processes of a color image. For a grayscale image, the color space transformation and the downsampling are not needed. Next, we introduce the three most important steps in JPEG compression process: the block splitting, the two-dimensional discrete cosine transform (2-D DCT), and the quantization. In this paper, our proposed schemes closely relate to these three steps.

In the step block splitting, the image is split into non-overlapped 8×8 blocks which are prepared for the next steps, because the DCT and the quantization must be performed in units of 8×8 block. Before computing the DCT, each coefficient in each 8×8 block falls in the range $[0, 255]$. But in order to reduce the dynamic range requirements in the following DCT processing, the range of each coefficient need to be shifted to $[-128, 127]$. After computing DCT of each block, all of these spatial domain coefficients are converted to frequency domain. If $x_{i,j}$ is a DCT coefficient of one block ($i, j \in [0, 7]$), then after DCT processing, the range of $x_{i,j}$ becomes $[xmin_{i,j}, xmax_{i,j}]$[1]. In each 8×8 DCT block, the coefficient in top-left corner is the DC (direct component) coefficient, and the remaining 63 coefficients are AC (alternating component) coefficients. The coefficients in upper left corner are the low frequencies, where the most important visual characteristics of the image are placed. In contrast, the highest frequencies are in the lower right corner and correspond to the details of the image. The quantization step allows to reduce the information. Each coefficient in each block is divided by the corresponding element in quantization table, and then rounding to the nearest integer. If $q(i, j)$ denotes the element of quantization table ($i, j \in [0, 7]$), the range of quantized coefficients is

$$\left[\text{round}\left(\frac{xmin_{i,j}}{q(i,j)}\right), \text{round}\left(\frac{xmax_{i,j}}{q(i,j)}\right)\right].$$

According to us, encryption can take place at three different steps during JPEG compression process (shown in Fig. 1). The first one encrypts the image before DCT in spatial domain. The second one encrypts image after DCT in frequency domain. The third one encrypts image after quantization.

Proposed Encryption Algorithm

In this paper, we propose three schemes which integrate encryption into JPEG compression process and compare them. These three schemes use the same encryption algorithm which is described in this section, and the three schemes are detailed in the sixth section.

Our encryption algorithm is symmetric. We use a pseudo-random function *prf* which takes a value n as input and returns a pseudo-random value in the range $[0, n[$. This function is associated with an initialization function prf_{init} that takes values *key* and *seed* as inputs, and allows to initialize the function *prf*.

The main idea is to take all the coefficients that have to be encrypted to construct a sequence. Then encrypt it using a pseudo-random sequence of the same length which is generated by function *prf*, and put the encrypted stream back.

There are three functions in our encryption algorithm:

Key generation: $key = KeyGen(\lambda)$ takes λ as input and returns a random value *key* of λ bits.

Encryption: $C = Enc(I, key, seed)$ uses *key* and a random value *seed* to initialize the pseudo-random function and generates a pseudo-random sequence. Then encrypts an image I using this sequence. The result is encrypted image C.

Decryption: $I' = Dec(C, key, seed)$ decrypts the encrypted image C using the pseudo-random sequence which is initialized by *key* and the random value *seed*.

In the following subsections, we detail the functions encryption and decryption.

Encryption

The plaintext image I is in JPEG format. First of all, the function $prf_{\text{init}}(key, seed)$ initializes the pseudo-random function *prf* using random values *key* and *seed*.

[1] For instance, if $i = 0, j = 0$, $xmin_{0,0} = -1024$ and $xmax_{0,0} = 1016$.

DOI: 10.2352/ISSN.2470-1173.2016.11.IMAWM-461

1. Suppose that xc denotes the coefficient we want to encrypt and that the range of xc is $[-xc_{\min}, +xc_{\max}]$. We note $n_{xc} = xc_{\min} + xc_{\max} + 1$.
2. Each coefficient xc is encrypted as follow:

$$e_{xc} = (((xc + xc_{\min}) + prf(n_{xc})) \mod n_{xc}) - xc_{\min} \quad (1)$$

From the result, we make sure that each encrypted coefficient e_{xc} falls in the range $[-xc_{\min}, +xc_{\max}]$.

Finally, by performing the above operations and the rest of JPEG compression processes, the encrypted image C in JPEG format is obtained.

Decryption

First of all, we use the function $prf_{\text{init}}(key, seed)$ with the two random values *key* and *seed* to initialize the pseudo-random function *prf*.

1. Each encrypted coefficient e_{xc} falls in the range $[-xc_{\min}, +xc_{\max}]$. We note $n_{xc} = xc_{\min} + xc_{\max} + 1$.
2. e_{xc} is decrypted as follow:

$$\begin{aligned} d_{xc} =& (((e_{xc} + xc_{\min}) + (n_{xc} - prf(n_{xc}))) \mod n_{xc}) \\ & - xc_{\min} \end{aligned} \quad (2)$$

Finally, by performing the above operations and the rest of JPEG compression processes, the decrypted image I' is obtained.

We can prove that, the decrypted image I' is equal to original image I, in other words, for all xc, we have $d_{xc} = xc$. The proof is as follow:

Proof.

$$\begin{aligned} d_{xc} =& (((e_{xc} + xc_{\min}) + (n_{xc} - prf(n_{xc}))) \mod n_{xc}) - xc_{\min} \\ =& ((((((xc + xc_{\min}) + prf(n_{xc})) \mod n_{xc}) - xc_{\min} + xc_{\min}) \\ & + (n_{xc} - prf(n_{xc}))) \mod n_{xc}) - xc_{\min} \\ =& ((xc + xc_{\min} + prf(n_{xc}) + n_{xc} - prf(n_{xc})) \mod n_{xc}) \\ & - xc_{\min} \\ =& ((xc + xc_{\min} + n_{xc}) \mod n_{xc}) - xc_{\min} \\ =& ((xc + xc_{\min}) \mod n_{xc}) - xc_{\min} \end{aligned} \quad (3)$$

Because $xc \in [-xc_{\min}, xc_{\max}]$ and $n_{xc} = xc_{\min} + xc_{\max} + 1$, therefore, $xc + xc_{\min} \in [0, n_{xc}[$. Then we have $(xc + xc_{\min}) \mod n_{xc} = xc + xc_{\min}$.

Thus we can prove that $d_{xc} = xc + xc_{\min} - xc_{\min} = xc$. □

Security Analysis of the Encryption Algorithm

Indistinguishability is an important property of security. Security in terms of indistinguishability depends on the capabilities of the adversary. The two most commonly used security definitions are Indistinguability under chosen plaintext attack (IND-CPA) and Indistinguability under chosen ciphertext attack (IND-CCA). In the chosen plaintext attack, the adversary has access to an encryption oracle whereas in the chosen ciphertext attack, he has also access to a decryption oracle.

In this section, we first remind some security definitions, and then we prove that our encryption algorithm is IND-CPA secure.

Security Definitions

IND-CPA [12] for a symmetric cryptosystem is represented by the game between an adversary and a challenger. Let $\mathscr{B}$ be the challenger and $\mathscr{A}$ be the adversary. Here is the game:

1. $\mathscr{B}$ generates a symmetric key k.

2. $\mathscr{A}$ may perform any number of encryptions and other operations on the result.

3. $\mathscr{A}$ sends two different plaintexts m_0 and m_1 to $\mathscr{B}$.

4. $\mathscr{B}$ chooses one bit $b \in \{0,1\}$ randomly, and sends the challenge $c = E_k(m_b)$ to $\mathscr{A}$. $E_k(m_b)$ is the encryption of the message m_b with the key k.

5. $\mathscr{A}$ can perform other operations and give a guess for the value of b, 0 or 1, to $\mathscr{B}$.

The adversary is modeled by a probabilistic polynomial time Turing machine. That means the adversary must complete the game and output a guess within a polynomial number of time steps. A cryptosystem is indistinguishable under chosen plaintext attack if the adversary wins the game with a negligible advantage. It means that he wins the game with a probability equals to $\frac{1}{2} + \varepsilon(\lambda)$, where $\varepsilon(\lambda)$ is a negligible function in the security parameter λ (i.e. the size of the key in bits).

A cryptosystem is IND-CPA$ secure if it has pseudorandom ciphertexts in the presence of chosen plaintext attack. The game representation of this security definition is the same as the game for IND-CPA except for steps **3.** and **4.** which become:

3. $\mathscr{A}$ sends one plaintext m to $\mathscr{B}$.

4. $\mathscr{B}$ chooses one bit $b \in \{0,1\}$ randomly. If he chooses 0, he sends $c = E_k(m)$ to $\mathscr{A}$. Else, he sends a random number of n bits. n is equal to the size of $E_k(m)$.

In other words, a cryptosystem is IND-CPA$ secure if the adversary cannot distinguish between a ciphertext and a sequence of random numbers with a probability superior to $\frac{1}{2} + \varepsilon(\lambda)$.

An IND-CPA$ secure cryptosystem is clearly IND-CPA secure. If an adversary cannot distinguish between the challenge c and a sequence of random numbers with a probability superior to $\frac{1}{2} + \varepsilon(\lambda)$, then he cannot guess the right value of b with a better probability.

Security of the Encryption Algorithm

In our encryption algorithm, we use prf_{init} and *prf* functions to constitute a Pseudo Random Number Generator (PRNG). The prf_{init} function initializes the PRNG and *prf* permits to have pseudo random numbers from a long sequence of pseudo random bits thanks to prf_{init}.

Suppose that our PRNG is secure, which means that we cannot distinguish between the generated sequence of pseudo random bits and a sequence of real random bits of the same size.

In our encryption algorithm, we initialize the PRNG with $prf_{\text{init}}(key, seed)$ where *key* and *seed* are real random numbers. Then, for each coefficient xc of the image, we encrypt it as Eq. (1), e_{xc} is the encrypted coefficient.

Theorem. *If prf is secure then our encryption algorithm is IND-CPA secure.*

Proof. If *prf* is a secure PRNG, we cannot distinguish between $prf(n_{xc})$ and a real random number according to the definition of

DOI: 10.2352/ISSN.2470-1173.2016.11.IMAWM-461

a secure PRNG. Therefore, we can replace $prf(n_{xc})$ by a real random number r in Eq. (1), and the encryption equation becomes $e_{xc} = (((xc + xc_{\min}) + r) \mod n_{xc}) - xc_{\min}$ where r is a real random number. This change affects the algorithm in a negligible manner.

Operations based on a real random number give a random number as result. We have $e_{xc} = r'$ where r' is a random number. This change does not affect the algorithm.

We proved that, if *prf* is secure, encrypting a coefficient is equivalent to return a random number. That is the definition of IND-CPA$ secure. So our encryption algorithm is IND-CPA$ secure.

Besides, if an encryption is IND-CPA$ secure, then it is IND-CPA secure. Therefore, our encryption algorithm is IND-CPA secure. □

Three Encryption Schemes and Experimental Results

We implement our encryption algorithm at three different steps of the JPEG compression process. The first one encrypts image before DCT in spatial domain. The second one encrypts image after DCT in frequency domain. The third one encrypts image after quantization. We detail them in this section, and experimental results are presented to compare the three schemes. We use LibJPEG [14] to encode and decode JPEG image, and all experiments were implemented in C/C++. As examples, we first take the grayscale and the color image "Lena" of 512×512 pixels to do a detailed experiment, and then we take 10 grayscale images and 10 color images of different sizes to do more tests.

There are three components of color image: one luminance component (Y) and two chrominance components (Cb, Cr). We encrypt these three components respectively and the method to encrypt each of them is the same as to encrypt a grayscale image. Therefore, in the next subsections, we concisely present how to encrypt a grayscale image. In all experiments, we choose two quantization tables to compress image with compression ratio Q=71 and Q=100 (as shown in Fig. 2). We compare their runtime, Peak Signal to Noise Ratio (PSNR) and Universal Image Quality Index (UIQI) [15], which are techniques to measure the image quality. PSNR is defined via the Mean Squared Error (MSE), and it is worth mentioning that in this paper, for color images the MSE is calculated by the sum of all squared value differences divided by image size and then by three. For the UIQI calculation of color image, we first convert the color image to grayscale image, and then use the defined formula to calculate its UIQI value. All the results of "Lena" are summarized in Tab. 1 and Tab. 2, the average results of other images are explained in Tab. 3.

$$\begin{bmatrix} 9 & 6 & 6 & 9 & 14 & 23 & 30 & 35 \\ 7 & 7 & 8 & 11 & 15 & 34 & 35 & 32 \\ 8 & 8 & 9 & 14 & 23 & 33 & 40 & 32 \\ 8 & 10 & 13 & 17 & 30 & 50 & 46 & 36 \\ 10 & 13 & 21 & 32 & 39 & 63 & 60 & 45 \\ 14 & 20 & 32 & 37 & 47 & 60 & 66 & 53 \\ 28 & 37 & 45 & 50 & 60 & 70 & 70 & 59 \\ 42 & 53 & 55 & 57 & 65 & 58 & 60 & 57 \end{bmatrix}$$

(a) $Q = 71$.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(b) $Q = 100$.

Figure 2. *Quantization tables of grayscale image or luminance component of color image.*

Notice that the plaintext images are already in JPEG format, so we have to first decompress the image to retrieve the bitmap.

Encryption Before DCT

In this scheme, we first decompress the image just after the reverse DCT. The range of coefficients we want to encrypt is $[-128, 127]$. We encrypt all the coefficients of the image using the encryption algorithm given in the fourth section, and then apply the 2-D DCT to each encrypted block. Then the DCT block is quantized with $Q = 71$ or $Q = 100$ and entropy encoded. The encrypted image is in JPEG format. During decryption, we decompress encrypted image just after the reverse DCT, and decrypt all coefficients. Fig. 3 shows the resulting encrypted and decrypted images with $Q = 71$. A part of decrypted image is zoomed, and we notice that there are some little specks distributed in the images. According to our experiments of other 20 images, we find that there are a large number of specks in some images.

As shown in Tab. 1 and Tab. 2, if Q=100, the decryption algorithm can reconstruct the plaintext grayscale image with PSNR of 55 dB and with UIQI of 0.9912, and reconstruct the plaintext color image with PSNR of 49.9 dB and with UIQI of 0.991. They are very high values which mean the decrypted images are almost the same as the plaintext images. For $Q = 71$, the value of PSNR is 33 dB for grayscale image and 29.6 dB for color image, which means there are perceived distortions, but the quality is acceptable. The value of UIQI is 0.7054 for grayscale image and 0.6834 for color image, which means based on the human visual system (HVS), the decrypted image is similar to the original one, but has some distortions. If the original image is compressed with compression ratio $Q = 71$, the PSNR value is 47.7 dB for grayscale image and 38.4 dB for color image, the UIQI value is 0.9833 for grayscale image and 0.9154 for color image. Comparing them with the values of PSNR and UIQI after decryption, there is a gap but not too large. The runtime of this scheme is around 60∼90 ms for grayscale image and 150∼200 ms for color image on an Intel i7 laptop, which is already a very efficient encryption.

We do more analyses for 10 grayscale images and 10 color images using this scheme, the average values of PSNR and UIQI are given in Tab. 3. If $Q = 100$, the average PSNR value of 10 grayscale images is 29.3 dB, which is not a high value, but the average UIQI value is 0.9026, meaning that the visual quality of these images is high. For 10 color images the average PSNR value is 35.4 dB and the average UIQI value is 0.9307, which means the visual quality is high. If $Q = 71$, for 10 grayscale images the average PSNR value is 16.4 dB, which is extremely poor, and the average UIQI value is 0.4283 which is only acceptable. For 10 color images the PSNR value is 18.8 dB and the UIQI value is 0.4641, which also means the quality is just acceptable. If the original image is only compressed with compression ratio $Q = 71$, the average PSNR value is 45.1 dB for 10 grayscale images and is 42.3 dB for 10 color images; the average UIQI value is 0.9561 for 10 grayscale images and is 0.9245 for 10 color images. Comparing them with the value of PSNR and UIQI after decryption, the gap is large. So we conclude that this scheme cannot reconstruct the plaintext image with a high quality.

Encryption After DCT

In this scheme, we first decompress the image just before the reverse DCT. The image is split into 8×8 blocks and the range of DCT coefficients we want to encrypt is $[xmin_{i,j}, xmax_{i,j}]$, where

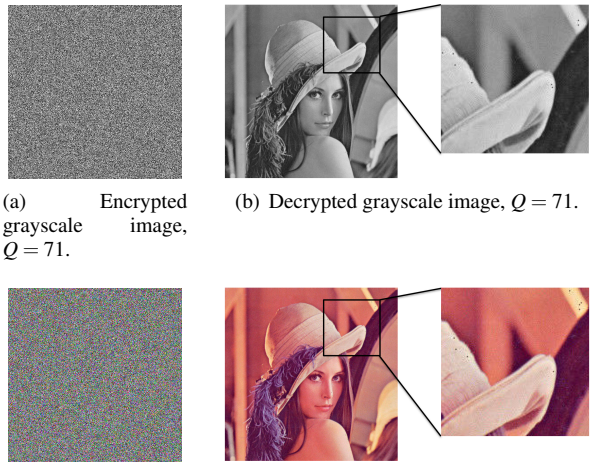
DOI: 10.2352/ISSN.2470-1173.2016.11.IMAWM-461

Tab. 1. Summary of the results for grayscale image "Lena".

	Q	Runtime (ms)		PSNR (dB)		UIQI	
		Encryption	Decryption	Compression without encryption	After decryption	Compression without encryption	After decryption
Encryption before DCT	71	89	64	47.7	33	0.9833	0.7054
	100	71	61	Inf	55	1	0.9912
Encryption after DCT	71	35	32	47.7	34.7	0.9833	0.8715
	100	31	31	Inf	35.1	1	0.9313
Encryption after quantization	71	19	10	47.7	35	0.9833	0.9217
	100	29	13	Inf	35.1	1	0.9313

Tab. 2. Summary of the results for color image "Lena".

	Q	Runtime (ms)		PSNR (dB)		UIQI	
		Encryption	Decryption	Compression without encryption	After decryption	Compression without encryption	After decryption
Encryption before DCT	71	186	171	38.4	29.6	0.9154	0.6834
	100	198	151	Inf	49.9	1	0.991
Encryption after DCT	71	73	65	38.4	33.3	0.9154	0.8279
	100	82	64	Inf	35	1	0.8951
Encryption after quantization	71	44	21	38.4	34.4	0.9154	0.8752
	100	25	22	Inf	35	1	0.8951



(a) Encrypted grayscale image, $Q = 71$.

(b) Decrypted grayscale image, $Q = 71$.

(c) Encrypted color image, $Q = 71$.

(d) Decrypted color image, $Q = 71$.

Figure 3. *Experimental results of encryption before DCT.*

$x_{i,j}$ is the DCT coefficients of one block ($i, j \in [0,7]$). The encryption is implemented block by block and each block is represented as a matrix. Remember that the human eye is more sensitive to lower frequencies than to higher frequencies [13]. So we only encrypt the DC coefficient and the first 14 AC coefficients according to "zigzag" order and set the remaining coefficients to 0 as described in matrix (4). In this way, we reduce the calculation and ensure that the attackers can only obtain encrypted coefficients.

Then the encrypted DCT block is quantized with $Q = 71$ or $Q = 100$ and entropy encoded. The encrypted image is in JPEG format. During decryption, we decompress encrypted image just before the reverse DCT, and only decrypt the DC coefficient and the first 14 AC coefficients according to the "zigzag" order block by block. Fig. 4 shows the resulting encrypted and decrypted images. A part of decrypted image is zoomed, and we find that there is no visible noise distributed in the images. For other 20 images, we find few specks in some images.

$$\begin{bmatrix} e_{dc} & e_{ac1} & e_{ac5} & e_{ac6} & e_{ac14} & 0 & 0 & 0 \\ e_{ac2} & e_{ac4} & e_{ac7} & e_{ac13} & 0 & 0 & 0 & 0 \\ e_{ac3} & e_{ac8} & e_{ac12} & 0 & 0 & 0 & 0 & 0 \\ e_{ac9} & e_{ac11} & 0 & 0 & 0 & 0 & 0 & 0 \\ e_{ac10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4)$$

As shown in Tab. 1 and Tab. 2, if compression ratio is $Q = 71$, the decryption algorithm can reconstruct the plaintext grayscale image with PSNR of 34.7 dB and with UIQI of 0.8715, and reconstruct the plaintext color image with PSNR of 33.3 dB and with UIQI of 0.8279. They are higher than the PSNR and UIQI values of first scheme with $Q = 71$, which means the decrypted image is more similar to the plaintext image. If $Q = 100$, the value of PSNR is 35.1 dB for grayscale image and 35 dB for color image, the value of UIQI is 0.9313 for grayscale image and 0.8951 for color image. They are high values which mean the decrypted image is very similar to the plaintext image. The gap between the values of PSNR and UIQI after compression without encryption and the values after decryption is smaller than the first scheme. The runtimes of encryption and decryption are around 30 ms for grayscale image and 60∼80 ms for color image, it takes less time than the encryption before DCT. So this scheme is more efficient

DOI: 10.2352/ISSN.2470-1173.2016.11.IMAWM-461

than the first one.

We do more analyses for 10 grayscale images and 10 color images using this scheme, the average values of PSNR and UIQI are given in Tab. 3. If $Q = 100$, for 10 grayscale images the average PSNR value is 30.9 dB and the average UIQI value is 0.8966; for 10 color images the average PSNR value is 33.2 dB and the average UIQI value is is 0.8937. It means that the decrypted image is similar to the original one, but has some distortions. If $Q = 71$, for 10 grayscale images the average PSNR value is 28.2 dB and the average UIQI value is 0.72; for 10 color images the average PSNR value is 30.1 dB and the average UIQI value is 0.7059. They are higher than the PSNR and UIQI values of first scheme. The gap between the values of PSNR and UIQI after compression without encryption and the values after decryption is smaller than the first scheme. So we can conclude that this scheme can reconstruct the plaintext image with a higher quality than the first one.

(a) Encrypted grayscale image, $Q = 71$.

(b) Decrypted grayscale image, $Q = 71$.

(c) Encrypted color image, $Q = 71$.

(d) Decrypted color image, $Q = 71$.

Figure 4. *Experimental results of encryption after DCT.*

Encryption After Quantization

In this scheme, we first decompress the image just before the dequantization and re-quantize it with $Q = 71$ and $Q = 100$. The image is split into 8×8 blocks and the range of quantized DCT coefficients we want to encrypt is

$$\left[\text{round}\left(\frac{xmin_{i,j}}{q(i,j)}\right), \text{round}\left(\frac{xmax_{i,j}}{q(i,j)}\right)\right],$$

where $q(i,j)$ is the element of quantization table ($i, j \in [0,7]$). The encryption is implemented block by block and each block is represented as a matrix. As for the second scheme, we only encrypt the DC coefficient and the first 14 AC coefficients according to "zigzag" order and set the remaining coefficients to 0. Then the encrypted quantized DCT block is entropy encoded. The encrypted image is in JPEG format. During decryption, we decompress encrypted image just before the dequantization, and only decrypt the DC coefficient and the first 14 AC coefficients according to the "zigzag" order block by block. Notice that, when the compression ratio $Q = 100$, this scheme is the same as the second one. Fig. 5 shows the resulting encrypted and decrypted images. A part of decrypted image is zoomed, and we find that there is no visible noise distributed in the images. For other 20 images, we have the same conclusion.

As shown in Tab. 1 and Tab. 2, if compression ratio is $Q = 71$, the decryption algorithm can reconstruct the plaintext grayscale image with PSNR of 35dB and with UIQI of 0.9217, and reconstruct the plaintext color image with PSNR of 34.4 dB and with UIQI of 0.8752, which are higher than the first and the second ones. That means the decrypted image is the most similar to the plaintext image. If Q=100, the results are the same as in second scheme. The gap between the values of PSNR and UIQI after compression without encryption and the values after decryption is the smallest one. For $Q = 71$, the runtime of encryption and decryption are 19 ms and 10 ms for grayscale image and 44 ms and 21 ms for color image, it takes a little less time than the encryption after DCT. For $Q = 100$, even though it has the same encrypted result as the second method, the runtime of encryption and decryption are less than the second one, which are 29 ms and 13 ms for grayscale image and 25 ms and 22 ms for color image. So this scheme is the most efficient one.

We do more analyses for 10 grayscale images and 10 color images using this scheme, the average values of PSNR and UIQI are given in Tab. 3. If $Q = 100$, the average PSNR and UIQI values are the same as in the second scheme. If $Q = 71$, for 10 grayscale images the average PSNR value is 30.8 dB and the average UIQI value is 0.8598; for 10 color images the average PSNR value is 33 dB and the average UIQI value is 0.8469. They are higher than the PSNR and UIQI values of first and second schemes, which means the decrypted image is the most similar to the plaintext image. The gap between the values of PSNR and UIQI after compression without encryption and the values after decryption is the smallest. So we can conclude that this scheme is the best one to reconstruct the plaintext image.

(a) Encrypted grayscale image, $Q = 71$.

(b) Decrypted grayscale image, $Q = 71$.

(c) Encrypted color image, $Q = 71$.

(d) Decrypted color image, $Q = 71$.

Figure 5. *Experimental results of encryption after quantization.*

Compared with [9], our encryption schemes have better performances.

DOI: 10.2352/ISSN.2470-1173.2016.11.IMAWM-461

Tab. 3. Summary of the results for 20 other images.

	Q	Grayscale image		Color image	
		Average PSNR (dB)	Average UIQI	Average PSNR (dB)	Average UIQI
Compression without encryption	71	45.1	0.9561	42.3	0.9245
	100	Inf	1	Inf	1
Encryption before DCT	71	16.4	0.4283	18.8	0.4641
	100	29.3	0.9026	35.4	0.9307
Encryption after DCT	71	28.2	0.72	30.1	0.7059
	100	30.9	0.8966	33.2	0.8937
Encryption after quantization	71	30.8	0.8598	33	0.8469
	100	30.9	0.8966	33.2	0.8937

Conclusion and Future Work

In this paper, we propose three JPEG image encryption schemes and compare them. They use the same encryption algorithm but are integrated into different steps of the JPEG compression process. All the encrypted results are JPEG images and the experiments show the decryption algorithm can reconstruct a high quality image. The security of our encryption algorithm is proved in this paper as well.

We are currently working on uploading the encrypted images to different platforms, e.g. Facebook, Instagram, Flickr, ..., to verify whether the encrypted images can be accepted by these platforms as correct image format. We will also try to improve our encryption algorithm to provide the decrypted image with a higher quality.

References

[1] Pennebaker, William B., and Joan L. Mitchell. JPEG: Still image data compression standard. Springer Science & Business Media, 1993.

[2] Dang, Philip P., and Paul M. Chau. "Image encryption for secure internet multimedia applications." Consumer Electronics, IEEE Transactions on 46.3 (2000): 395-403.

[3] Dey, Shuvashis. "SD-AEI: An advanced encryption technique for images." Digital Information Processing and Communications (ICDIPC), 2012 Second International Conference on. IEEE, 2012.

[4] Zhu, Zhi-liang, et al. "A chaos-based symmetric image encryption scheme using a bit-level permutation." Information Sciences 181.6 (2011): 1171-1186.

[5] Usman, Koredianto, et al. "Medical image encryption based on pixel arrangement and random permutation for transmission security." e-Health Networking, Application and Services, 2007 9th International Conference on. IEEE, 2007.

[6] Luo, Yuling, Minghui Du, and Dong Liu. "JPEG Image Encryption Algorithm Based on Spatiotemporal Chaos." Chaos-Fractals Theories and Applications (IWCFTA), 2012 Fifth International Workshop on. IEEE, 2012.

[7] Korshunov, Pavel, and Touradj Ebrahimi. "Scrambling-based tool for secure protection of JPEG images." Image Processing (ICIP), 2014 IEEE International Conference on. IEEE, 2014.

[8] Qiao, Lintian, and Klara Nahrstedt. "Comparison of MPEG encryption algorithms." Computers & Graphics 22.4 (1998): 437-448.

[9] Ahmed, Fawad, M. Y. Siyal, and Vali Uddin Abbas. "A perceptually scalable and jpeg compression tolerant image encryption scheme." Image and Video Technology (PSIVT), 2010 Fourth Pacific-Rim Symposium on. IEEE, 2010.

[10] Backes, Michael, et al. "X-pire 2.0: a user-controlled expiration date and copy protection mechanism." Proceedings of the 29th Annual ACM Symposium on Applied Computing. ACM, 2014.

[11] Lian, Shiguo, Jinsheng Sun, and Zhiquan Wang. "A novel image encryption scheme based-on JPEG encoding." Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on. IEEE, 2004.

[12] Marcedone, Antonio, and Claudio Orlandi. "Obfuscation $\Rightarrow$ (IND-CPA Security $\not\Rightarrow$ Circular Security)." Security and Cryptography for Networks. Springer International Publishing, 2014. 77-90.

[13] Fonteneau, C., J. Motsch, and M. Babel. et O. Déforges,"A Hierarchical Selective Encryption Technique in a Scalable Image Codec,". Proc. of International Conference in Communications.

[14] Independent JPEG Group. "Libjpeg 6b." URL http://www.ijg.org March (1998).

[15] Wang, Zhou, and Alan C. Bovik. "A universal image quality index." Signal Processing Letters, IEEE 9.3 (2002): 81-84.

Author Biography

Kun He received her B.E. in telecommunication from the Xidian University, China (2011) and her M.Sc in telecommunication from Télécom Bretagne, France (2013). Since then she has begun studying for her Ph.D. at B-Com in Rennes, France. Her thesis is focused on the image privacy in public image-sharing platforms.