https://doi.org/10.1016/j.jvcir.2017.01.021

Joint image compression and encryption based on order-8 alternating transforms

Peiya Li, Kwok-Tung Lo

Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China

Abstract

In this paper, we propose a novel joint image compression and encryption scheme based on JPEG standard. We realize image encryption at JPEG's transformation stage. Instead of only using 8×8 discrete cosine transform (DCT) for transformation, we generate new orthogonal transforms by embedding an extra rotation angle of π to different stages' butterflies in 8×8 DCT's flow-graph, and then apply them alternatively for transformation according to a predefined secret key. By carefully controlling the number of rotation angles embedded, the quality control of encrypted images can also be achieved. The encryption algorithm is further enhanced by performing block permutation before the entropy encoding stage. Extensive experiments have been conducted to show the good protection and compression performance of our encryption schemes. Finally, a detailed security analysis is provided to show the encryption schemes' resistance to various cryptanalysis methods, such as brute-force attack, key sensitivity analysis, replacement attack and statistical attack.

Email addresses: yolanda.peiya@connect.polyu.hk (Peiya Li), kwok.tung.lo@polyu.edu.hk (Kwok-Tung Lo)

Preprint submitted to Vis. Commun. Image R. *October 26, 2016*

Keywords: Image encryption, orthogonal transforms, security analysis, JPEG standard

1. Introduction

In recent years, with the rapid development of multimedia technologies, many powerful and interesting new applications have been developed for people to share their images stored in mobile smart devices through different social network platforms, such as Facebook and Instagram. In typical use of images, owners tend to store them for future use or distribute them to the specific people through Internet which is particularly vulnerable to eavesdropping and intercepting, thus there exists a great demand on efficient, secure image storage and transmission. Encryption is one of the common ways to ensure image security.

Until now, there are various cryptographic techniques, like Data Encryption Standard (DES) and Advanced Encryption Standard (AES) [1], but they are designed primarily for text data security. For image encryption using these conventional algorithms, the computational cost will be high because of the large size of image data. Moreover, unlike text data, the information density of images is much lower, and the real-time processing is often required for many image applications, thus an encryption strategy with low computational cost and fast encryption/decryption speed is desired for image safety.

In the past time, scrambling techniques were widely used by researchers to achieve image encryption. These techniques include simple permutation operations and affine transformation operations in the time or frequency domain [2–5]. However, along with the rapid increasing of computing power, breaking this kind of encryption schemes becomes much easier. Then, considering the fact that compression is a must-do step for most images we see on the Internet, the focus of image security research shifted to integrating image compression process with encryption, for the purpose of reducing encryption and decryption time in image communication and processing. The most popular method for this encryption direction is partial encryption or selective encryption, which applies encryption to a portion of coefficients from either the final results or the intermediate stages of the image compression system. The major aim of partial image encryption is to reduce the amount of data for encryption while obtaining a certain level of security.

Based on the different processing order of encryption and compression process, partial image encryption can be divided into three categories: pre-compression, in-compression, and post-compression [6]. Pre-compression means performing encryption before compression, and decompression before decryption. In [7], Tang proposed a method to realize partial image encryption by encrypting the DC coefficients with DES and randomly permuting the AC coefficients rather than the standard zigzag scanning order. This encryption scheme is format compliant to JPEG, but it introduces about 40% loss in compression efficiency, because the new permutation disrupts the probability distribution of run-lengths, and renders the performance of Huffman tables less than optimal [8]. For in-compression category, image encryption/decryption and image compression/decompression are performed jointly. In [9–11], Wu *et al.* realized partial encryption in the entropy coding stage by using multiple Huffman tables alternately in a secret order. This encryption scheme can achieve very high-visual degradation without sacrificing the compression performance, but it is not format compliant, because the decoder needs to decrypt the Huffman table used for encryption in order to do decompression. Post-compression algorithms perform encryption after compression, and are generally compression friendly; no modifications are needed for encryption and decryption at the encoder or decoder side. In [12], they proposed a JPEG image encryption method by only encrypting those bits that are used to fully specify the signs and magnitudes of nonzero coefficients in the Huffman coding stage. This algorithm can obtain high-visual degradation and is format compliant to JPEG, but it is not tunable since its encryption parameters are static.

In [13–16], Au-Yeung *et al.* proposed to realize partial video encryption by embedding the encryption scheme at the transformation stage during the encoding process. Since transformation is a must-do step in the video encoder and decoder, an obvious advantage of this scheme is that it introduces nearly zero extra computations. They generated a number of new orthogonal transforms with similar coding efficiency to DCT-II by modifying angle values in DCT-II's flow-graph structure, and then applied these new transforms alternately according to a predefined secret key to realize partial video encryption. In [16], they first suggested a simple design to generate new orthogonal transforms by selecting four different sets of rotation angles ($\theta_1 = \pi/4$, $\theta_2 = 3\pi/8$), ($\theta_1 = 3\pi/8$, $\theta_2 = \pi/4$), ($\theta_1 = 7\pi/24$, $\theta_2 = 8\pi/24$), ($\theta_1 = 8\pi/24$, $\theta_2 = 7\pi/24$), at the 2^{nd} stage of the 4-point DCT's flow-graph structure (shown in Fig. 1). Then in order to enlarge the differences among these alternative transforms which can offer a good protection against possible attacker, they achieved the maximum difference in [15] by allowing an extra rotation of π to θ_1 or/and θ_2. These newly generated transforms have exactly the same coding efficiency as DCT since they are the sign-flipped version of the original DCT. Next, they extended the 4×4 transforms based encryption frame-work into 8×8 case [14], which allows more room to do butterflies' sign-flipping and thus to embed the encryption. In [14], three different encryption algorithms were proposed, and they claimed that *Algorithm-3* could get the best balance between the encryption power and coding efficiency.

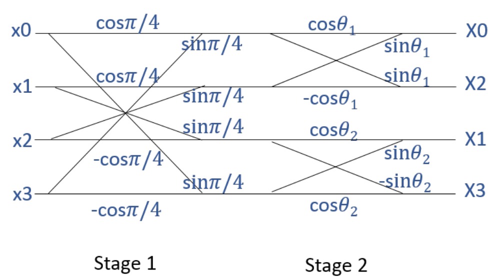


Figure 1: Flow graph of 4-point (1-D) DCT ($\theta_1 = \pi/4$, $\theta_2 = 3\pi/8$)

In our work, we propose a new joint image compression and encryption scheme with controllable image quality by embedding encryption algorithm at JPEG's transformation stage. The proposed scheme can achieve a sufficiently high level of security while maintain the good compression performance of JPEG, and it is format compliant to JPEG standard. Instead of using the 8×8 DCT alone for transformation, we develop new order-8 orthogonal transforms by adopting the sign-flipping strategy used in [14], but the butterflies we do sign-flips are different from [14], resulting in different new transform sets for transformation. Then, encryption algorithm using the newly generated transform sets is presented. Encryption and compression performances are evaluated through some common judgement criteria. And we have shown that our new orthogonal transforms have better coding efficiency than those generated in [14]. By carefully selecting the butterflies for angle rotation, we can control the visual quality of the encrypted images. Additionally, to better resist the direct replacement attack, we modify the encryption algorithm by adding the block permutation operation. A detailed security analysis of our proposed encryption scheme is provided finally to show its resistance to various attacks, such as brute-force attack, key sensitivity analysis, replacement attack, and statistical analysis.

In the rest of the paper, Section 2 introduces the method to generate new order-8 orthogonal transforms which will be used alternatively for transformation according to the encryption key in our encryption algorithm. Section 3 describes the encryption/decryption algorithm, evaluate and compare its performance with JPEG and Au-Yeung's *Algorithm-3* [14], together with the quality control realization of encrypted images. Section 4 shows the detailed security analysis results of our encryption algorithms. Section 5 gives a conclusion and presents our future research directions.

2. Generation of new order-8 orthogonal transforms

In JPEG, 8×8 DCT is used for transformation, because it has proved to be the best transform in terms of compression ability when the correlation among inter-pixels is strong — which is actually true in most natural pictures. In [17], a fast DCT-II algorithm was proposed in the form of matrices and was illustrated by a signal-flow graph. In our method for generating new orthogonal transforms, we modify the signal-flow graph structure through introducing extra rotation angle of π to different butterflies. Therefore, we first introduce the underlying fast computational algorithm for DCT-II, then explain how to generate new transforms based on it.

2.1. One fast computational algorithm for DCT-II

The fast DCT computational algorithm [17] is based upon matrix decomposition. In general, an order-N type II DCT matrix can be written into the following recursive form:

$$
\begin{aligned}
\left[C_N^{II}\right] &= [P_N] \begin{bmatrix} P_{N/2}^T C_{N/2}^{II} & 0 \\ 0 & R_{N/2} \end{bmatrix} [B_N] \,, \\
[B_N] &= \frac{\sqrt{2}}{2} \begin{bmatrix} I_{N/2} & \bar{I}_{N/2} \\ \bar{I}_{N/2} & -I_{N/2} \end{bmatrix},
\end{aligned}
\tag{1}
$$

where $[P_N]$ is an $N \times N$ permutation matrix which permutes the transformed vector from a bit reversed order to a natural order. $\left[I_{N/2}\right]$ is the identity matrix and $\left[\bar{I}_{N/2}\right]$ is the anti-diagonal identity matrix. $\left[R_{N/2}\right]$ can be decomposed into $(2\log_2 N - 3)$ matrices.

Since in JPEG standard, only order-8 DCT-II is used for transformation, here we present the flow-graph of it in Fig. 2, and its fast computational formula can be described as:

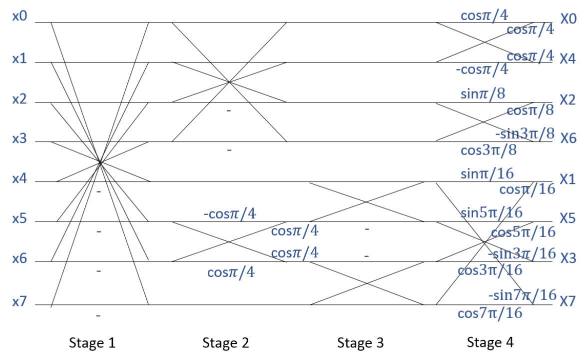


Figure 2: Flow graph of 8-point (1-D) DCT

$$
\begin{aligned}
\left[C_8^{II}\right] &= [P_8]\begin{bmatrix} P_4^T C_4^{II} & 0 \\ 0 & R_4 \end{bmatrix}[B_8], \qquad (2) \\
\left[C_4^{II}\right] &= [P_4]\begin{bmatrix} P_2^T C_2^{II} & 0 \\ 0 & R_2 \end{bmatrix}[B_4], \\
[P_4] &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
[R_2] &= [M1] = \begin{bmatrix} \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \\ -\sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} \end{bmatrix}, \\
[P_2] &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \left[C_2^{II}\right] = \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},
\end{aligned}
$$

$$[P_8] \;=\; \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$[R_4] \;=\; [M1]\,[M2]\,[M3]\,,$$

where

$$[M1] = \begin{bmatrix} \sin\frac{\pi}{16} & 0 & 0 & \cos\frac{\pi}{16} \\ 0 & \sin\frac{5\pi}{16} & \cos\frac{5\pi}{16} & 0 \\ 0 & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ -\sin\frac{7\pi}{16} & 0 & 0 & \cos\frac{7\pi}{16} \end{bmatrix},$$

Computation of $[M1]$ is equivalent to the butterfly denoted in the bottom half of Stage-4 in Fig. 2.

$$[M2] = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

Computation of $[M2]$ is equivalent to the butterfly denoted in the bottom half of Stage-3 in Fig. 2.

$$[M3] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\cos\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & \cos\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

Computation of $[M3]$ is equivalent to the butterfly denoted in the bottom half of Stage-2 in Fig. 2.

2.2. Generation of new orthogonal transforms

Considering the encryption and compression performance, the coding efficiency of the new generated transforms should either be exactly the same to what can be achieved by DCT or just fall slightly. Thus, we do not introduce angle rotation to butterflies in Stage-1 and Stage-2, because this will cause big changes in the transform coefficients, which will eventually affect the overall coding efficiency.

In our transform generation method, using the extra rotation angle of π, we generate two different sets of new orthogonal transforms, *TS1* and *TS2*. *TS1* has 16 different orthogonal transforms, generated by introducing sign-flips (an extra rotation angle of π)into the four butterflies at Stage-4 in Fig. 2, which can be described in the following mathematical formula:

$$\left[C_8^{II}\right] = [P_8]\,[T_1] \begin{bmatrix} P_4^T C_4^{II} & 0 \\ 0 & R_4 \end{bmatrix} [B_8]\,, \tag{3}$$

where $[T_1] = diag(\cos\theta_1, \cos\theta_1, \cos\theta_2, \cos\theta_2, \cos\theta_3, \cos\theta_4, \cos\theta_4, \cos\theta_3)$, $\theta_i =$ 0 or π $(i = 1, 2, 3, \text{or } 4)$. Here, the normalized coefficients are ignored. The coding efficiencies of transforms in *TS1* are exactly the same to that of DCT because they are just the sign-changed format of the original DCT elements.

TS2 has 64 different orthogonal transforms, generated by introducing sign-flips into the four butterflies at Stage-4 and the two butterflies at Stage-3 in Fig. 2, which can be described in the following mathematical formula:

$$\begin{aligned} \left[C_8^{II}\right] &= [P_8]\,[T_1]\begin{bmatrix} P_4^T C_4^{II} & 0 \\ 0 & R_4^{'} \end{bmatrix}[B_8]\,, \\ \left[R_4^{'}\right] &= [M1]\,[T_2]\,[M2]\,[M3]\,, \end{aligned} \tag{4}$$

where $[T_2] = diag(\cos\theta_5, \cos\theta_5, \cos\theta_6, \cos\theta_6)$, $\theta_i = 0$ or π ($i = 5$ or 6), and the normalized coefficients are ignored. The coding efficiency of these transforms will be a little lower than that of DCT because some of them will change not only coefficients sign, but also their magnitudes after transformation.

3. Joint image compression and encryption based on alternating transforms

3.1. Encryption algorithm and experiment results

The encryption scheme can be divided into two parts: 1) random (secret) key generation, and 2) alternating transforms according to the secret key. For the key generation, we use the RC4 algorithm, which turns to be one of the most commonly-used random key generators [18]. The encryption algorithm is stated as follows:

Encryption Algorithm-1

Step 1: Initialize the RC4 key generator with a predefined 128-bit key;

Step 2: For an input 8×8 image block, do

- *Step* 2.1: Get 52 bits from the random generator;
- *Step* 2.2: Use the first 4 bits to select one transform from *TS1* for *all* rows in the 1^{st} dimension;
- *Step* 2.3: Use the following 6×8 bits to select one transform from *TS2* for *each* column in the 2^{nd} dimension;

Step 3: Transform each 8×8 image block using the selected transforms, then perform JPEG's quantization and entropy coding procedure;

Step 4: Repeat *Step* 2 and *Step* 3 until all 8×8 blocks are processed.

In *Algorithm-1*, we use *TS1* and *TS2* together and implement the 1^{st} and 2^{nd} dimension transformation separately. These operations will change the transformed coefficients in both signs and magnitudes, thus cryptanalyzing our encrypted images through some sign-flips on the DCT transformed coefficients of the same data block is not feasible.

For the decryption algorithm, we just follow JPEG's decoding process by using the encryption key to select the corresponding transforms for de-transformation. In Fig. 3, we present the encrypted image and decrypted image of 'Lena', when the quality factor is 20. All images used in our experiment are taken from the USC-SIPI image database available on the website "http://sipi.usc.edu/database". We assume that the standard IDCT is always used for decryption when the encryption key is unknown. Performance comparison between our encryption algorithm and the *Algorithm-3* in reference [14] is shown in Fig.4. In Fig. 4(a), we can observe a large PSNR value drop when the encryption key is unknown and only IDCT is used for decryption, which means a good protection ability of our encryption algorithm. When the encryption key is known, the PSNR value, which reflects the coding efficiency of transforms, of *Algorithm-1* is more close to that of JPEG standard than that of the reference paper's algorithm, illustrating that our transform sets' coding efficiency is better than the transform sets in [14]. For the PSNR-BS (Bit-stream Size) relationship and PSNR-CR (Compression Ratio) relationship shown in Fig. 4(b) and Fig. 4(c), compared with JPEG standard, our algorithm also has better compression performance and smaller bit-stream size than *Algorithm-3* in [14].

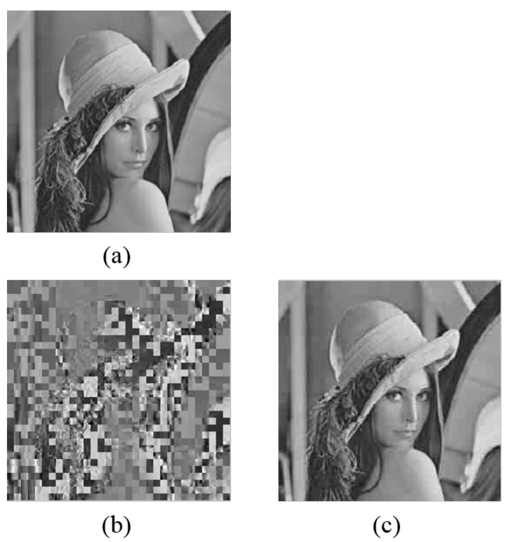
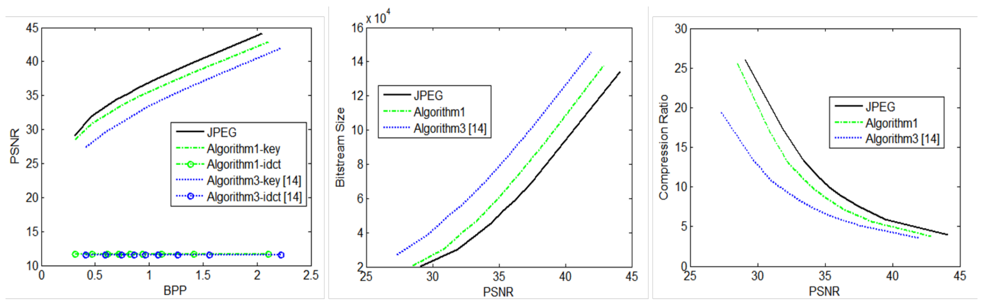


Figure 3: Performance of *Algorithm-1*: (a) plain Lena image, (b) encrypt image, (c) decrypt image



(a) (b) (c)

Figure 4: Performance comparison among different encryption algorithms: (a) BPP-PSNR relationship, (b) PSNR-BS relationship, (c) PSNR-CR relationship

3.2. Quality control of our encryption scheme

For partial image encryption, an encrypted image with poor visual quality can be generated without the encryption key. However, different applications may have different visual quality requirements of the encrypted images, thus it is necessary to allow the service provider a chance to control how bad the encrypted image quality will be.

In our encryption scheme, in order to achieve quality control, instead of changing all the four angles in matrix $[T_1]$, we select some of them to be π while keeping the others unchanged. Table 1 has listed the different PSNR performances of various selections of θ_i $(i = 1, 2, 3, \text{or } 4)$ for encryption. Three images are used for testing with quality factor to be 20. From Table 1, it is clearly shown that among four rotation angles, θ_1 has the biggest influence on the PSNR value, as it controls the DC component of each 8×8 block. In Fig. 5, we present the visual results of the three representative selections in Table 1 of these three images. It is clearly that the visual qualities of these images are quite different, which are consistent with their PSNR values. For example, images in Case (b) are visually much more pleasant than images in Case (c) and Case (d). Thus, we can choose whether to change θ_1 or not to obtain high or low encryption ability, and change other three rotation angles to make some fine adjustment on the quality of images.

Table 1: PSNR [dB] performance of various selection of θ_i for encryption

Angle change \ Image	Lena	Aerial	Peppers
None (a)	31.9051	28.3087	32.4234
Stage-3 (b)	22.8940	19.0769	23.7966
Stage-3 & θ_1	11.5145	10.5809	10.4267
Stage-3 & θ_2	21.2139	17.6990	22.2106
Stage-3 & θ_3	19.9361	17.5506	20.8673
Stage-3 & θ_4	22.4611	19.1113	23.4249
Stage-3 & $\theta_1\theta_2$	11.8844	10.4773	10.4633
Stage-3 & $\theta_1\theta_3$	11.9228	10.5364	10.4736
Stage-3 & $\theta_1\theta_4$	11.9304	10.5626	10.4745
Stage-3 & $\theta_2\theta_3$ (c)	18.9731	16.6755	20.1005
Stage-3 & $\theta_2\theta_4$	21.1142	17.4612	21.8641
Stage-3 & $\theta_3\theta_4$	19.7323	17.4131	20.6949
Stage-3 & $\theta_1\theta_2\theta_3$	11.8025	10.4472	10.4982
Stage-3 & $\theta_1\theta_2\theta_4$	11.7619	10.4329	10.5164
Stage-3 & $\theta_1\theta_3\theta_4$	11.8245	10.4690	10.5340
Stage-3 & $\theta_2\theta_3\theta_4$	18.5591	16.5285	20.0187
Stage-3 & $\theta_1\theta_2\theta_3\theta_4$ (d)	11.6248	10.3927	10.4607

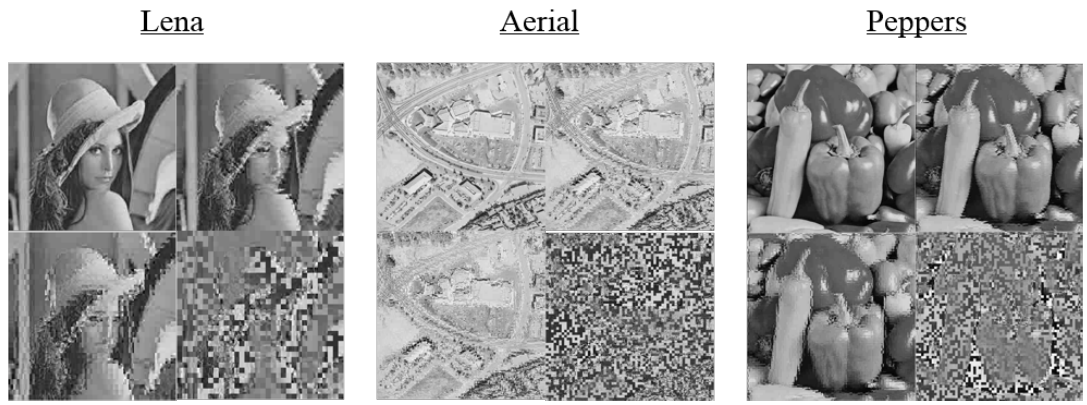


Figure 5: Encrypted images with different rotation angles change (left-top: Case (a) in Table 1; right-top: Case (b) in Table 1; left-bottom: Case (c) in Table 1; right-bottom: Case (a) in Table 1

3.3. Improved encryption scheme

In Fig. 3, we can observe that the encrypted image of 'Lena' under *Algorithm-1* still reveals some information about the original image. Thus in order to make encrypted image become more chaotic, we introduce the block permutation operation after the quantization procedure, which means that after we obtain all quantized 8×8 blocks, before doing the entropy coding stage, we first disrupt the order of these 8×8 blocks, according to the encryption key. The permutation algorithm we select is Fisher-Yates Shuffle which is used for generating a random permutation of a finite linear array [19]. To shuffle an array S of n elements, Fisher-Yates Shuffle do

for $i \leftarrow n$ to 2 **do**

  $j \leftarrow$ *random integer with* $1 \leq j \leq i$

  exchange $S[j]$ *and* $S[i]$

end for

When applying this shuffle algorithm in our encryption scheme, S is the original order of all 8×8 blocks, n is the number of 8×8 block, the random integer in each loop is obtained from the RC4 generated pseudo-random bit-stream, which can be described as following:

Random Integer Generation

---

1. Chose an integer r satisfying $2^r \geq n$;
2. Obtain r bits from the RC4 generated key-stream and convert them to a number x such that $0 \leq x < 2^r$;
3. Compute $t = \lfloor \frac{x}{n} \rfloor$, and output $x - tn$ as the random integer;

We name the block permutation embedded encryption algorithm as *Algorithm-1'*. The corresponding encrypted Lena image is shown in Fig. 6. It is obvious that the encrypted Lena image of *Algorithm-1'* is more chaotic than that of the image encrypted by *Algorithm-1*. Moreover, because we perform the permutation in 8×8 block unit, the final bit-stream size will only change slightly, which means that the good encryption and compression performances of *Algorithm-1* can be maintained.

Figure 6: Lena image encrypted by *Algorithm-1'* (left: plain image; right: cipher image

4. Security analysis

4.1. Key space and encryption space

There are many possible cryptographic attacks during the decoding, such as known-plaintext attack, chosen-plaintext attack, and ciphertext-only attack. Among them, the ciphertext-only attack is the most realistic and basic one in which attackers can only obtain the encrypted data. Here we evaluate the effectiveness of our encryption algorithms under this type of attack.

One of the typical methods for ciphertext-only attack is to try all possible keys in the brute-force manner. In our algorithm, we apply the RC4 key generator with a 128-bit key, thus obtain a 2^{128} key space, which is not feasible for attackers to guess. However, instead of guessing the key we use, attackers can guess the transforms we use for encryption. If we define the encryption space to denote how many rotation angles have been embedded into butterflies, then the encryption space of each 8×8 block for *Algorithm-1* is 2^{52} (2^4 for all rows in the 1^{st} dimension, 2^6 for each column in the 2^{nd} dimension). Since there will be 1024 blocks with 8×8 size in an 256×256 image, it is not feasible to try all transforms for all 8×8 blocks. Therefore, adopting the brute-force method to recover our encrypted image is extremely difficult.

4.2. Key sensitivity analysis

According to Kerckhoff's principle, the security of an encryption system should rely on the secrecy of the encryption/decryption key instead of the encryption algorithm itself [20]. A good cryptosystem should be extremely sensitive with respect to the key used in the algorithm, which should satisfy the following two conditions to indicate a high key sensitivity level [21]:

1) The key space should be discretized in such a way that two ciphertexts encrypted by two slightly different encryption keys should be completely different;
2) The ciphertext should not be correctly decrypted even if there is a slight difference in the encryption and decryption key.

If an encryption scheme satisfies the above-mentioned conditions, then its key sensitivity is considered high. In our encryption scheme, we use RC4 to generate the pseudorandom keystream which is initialized with 128-bit data, thus the 128-bit data is considered as the secret key of our cryptosystem. Overall, RC4 randomizes an array of 256 elements called S, and its output at each stage is a random element selected from S. To generate the keystream, two processes are used in RC4: a key-scheduling algorithm (KSA), which is used to initialize the permutation of S, and a pseudo-random generation algorithm (PRGA), to select the random elements and modify the original permutation of S. The pseudo-codes for the two processes are described as follows:

Key-scheduling algorithm

---

for $i \leftarrow 0$ to 255 **do**

  $S[i] \leftarrow i$

end for

$j \leftarrow 0$

for $i \leftarrow 0$ to 255 **do**

$j \leftarrow (j + S[i] + key[i \mod keylength]) \mod 256$

exchange $S[i]$ *and* $S[j]$

end for

Pseudo-random generation algorithm

$i \leftarrow 0$

$j \leftarrow 0$

while *GeneratingOutput* **do**

$i \leftarrow (i + 1) \mod 256$

$j \leftarrow (j + S[i]) \mod 256$

exchange $S[i]$ *and* $S[j]$

$K \leftarrow S[(S[i] + S[j]) \mod 256]$

K *as output*

end while

In our scheme, we use 128-bit data as the key, totally 16 decimals ranged in [0,255], thus the *keylength* used in KSA is 16. It is obvious that any minor change in *key* will alter the initial permutation result of S, and eventually affect the output K, which determines the following new transforms selection process and 8×8 blocks' permutation result. Though we do not use the keystream to directly modify pixel values of the plainimage (like XOR operation between pixels and keystream bits), changes occurred in keystream will still greatly impact the final encrypted/decrypted images, which can be seen in the following two conditions' verification results.

To verify the first condition, we slightly modify the 16 decimals' *key* by giving an increment of 1 to the last decimal. The correlation coefficient between different images encrypted using original key and slightly different key is shown in Table 2. Moreover, in Fig. 7, we have taken 'Aerial' as an example to present the cipher images with slightly different encryption key and the difference image of *Algorithm-1* and *Algorithm-1'*. The low correlation coefficient and chaotic difference images prove the fulfilment of condition 1 of key sensitivity analysis for our proposed encryption scheme.

Table 2: Correlation coefficient for image encrypted with original key and slightly different key

Image \ Algorithm	*Algorithm-1*	*Algorithm-1'*
Lena	0.0351	0.0057
Aerial	0.0201	-0.0018
Peppers	-0.0090	0.0269
Clock	0.0050	-0.0364
Resolution chart	0.0229	0.0123
Chemical plant	-0.0068	0.0014
Couple	0.0161	-0.0010
Stream and bridge	0.0094	0.0006
Sailboat	0.0011	0.0121
Baboon	-0.0009	-0.0080

To test the second condition, the encrypted image corresponding to plain image is decrypted with a slightly different decryption key rather than the original one. In our encryption scheme, we use the symmetric key system,

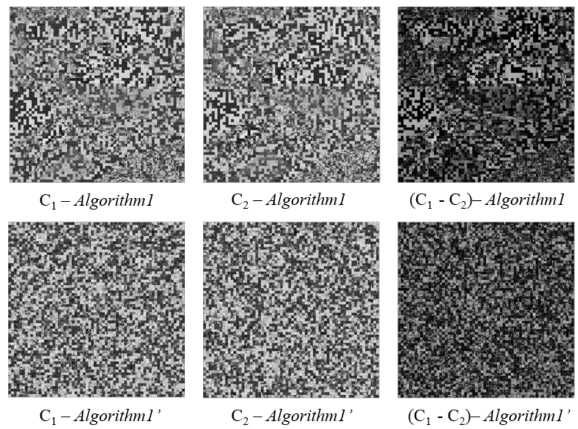
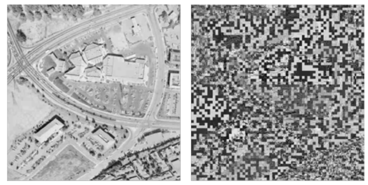


Figure 7: Key sensitivity analysis for encryption process — Encrypted 'Aerial' image with slightly modified encryption keys and their difference images

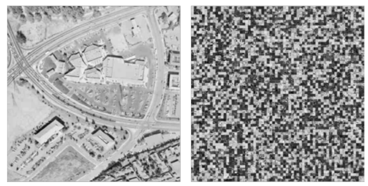
thus the encryption key and the decryption key are the same. We modify the decryption key by giving an increment of 1 to the last decimal of *key*. Decrypted images of 'Aerial' under *Algorithm-1* and *Algorithm-1'* obtained with slightly different decryption keys are shown in Fig. 8. It is clear that even when there is a small variation in the decryption key, the correct decryption cannot be realized under our encryption scheme. Thus, our proposed technique also satisfies the second condition of key sensitivity analysis.

4.3. Security against replacement attack

Replacement attacks are used to break multimedia encryption algorithms, which try to recover the plain media by replacing the encrypted parameter with the unencrypted ones or some others [22]. It can be divided into two categories: direct replacement and correlation-based replacement. Direct replacement means to reconstruct the plain media content by replacing some of the encrypted data with other ones under the condition of knowing only



(a) *Algorithm-1*

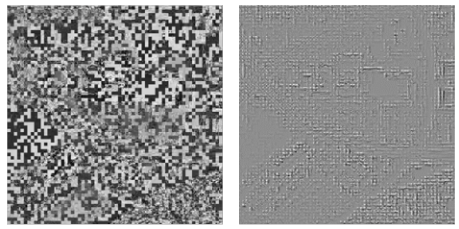


(b) *Algorithm-1'*

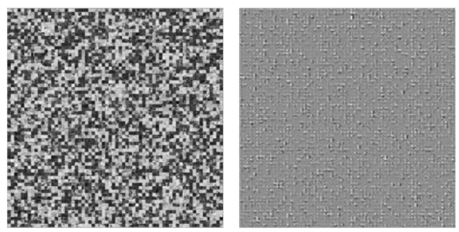
Figure 8: Key sensitivity analysis for decryption process: left image decrypted with right encryption key, right image decrypted with slightly modified decryption key

the cipher media content. Correlation-based replacement is similar to direct replacement with the difference that some of the encrypted data is replaced by unencrypted data.

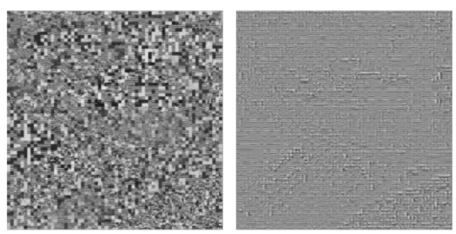
In our proposed image encryption scheme, we realize encryption at JPEG's transformation stage. Instead of choosing important elements to be encrypted, and leaving unimportant ones to be unencrypted, we do not give them a significance level, but encrypt all of them together by changing their signs or/and magnitudes. Thus the correlation-based replacement attack is infeasible for attacking our scheme. We analyse our technique's security against the direct replacement attack by assigning all dc coefficients to 128 and all ac coefficients to positive, and the corresponding data of the decrypted 'Aerial' images under our proposed two algorithms (*Algorithm-1*, *Algorithm-1'*) and *Algorithm-3* in [14] are shown in Fig. 9. We can observe that encryption with block permutation have better defense ability against the direct replacement attack than encryption without block permutation and *Algorithm-3* in [14], which both reveal some outline information about the plain Aerial image after the direct replacement attack operation.



(a) *Algorithm-1*



(b) *Algorithm-1'*



(c) *Algorithm-3* in [14]

Figure 9: Direct replacement attack analysis for different encryption algorithms (left: encrypt image, right: decrypt image)

4.4. Statistical model-based attack

Statistical model-based attack aims to recover the cipher image's intelligibility under the condition of knowing only cipher images. In this kind of attack, the degradation of the cipher image is reduced by using a statistical model [22]. The key step is to reconstruct an image with the cipher image's statistical model. If degradation of the cipher image is very strong, this attack will not work, which means that we can resist this type of attack by ensuring the low quality of our encrypted images, such as a small PSNR value.

Apart from the PSNR measure, attackers also can study the relationship between plain image and cipher image without knowing the encryption key to decrypt the cipher image. Generally, histograms and correlation diagrams of the original image and the encrypted image are two ways to indicate the degree of relationship between the plain image and cipher image [23–26]. Histogram analysis shows the distribution of pixels in the image by counting the number of each pixel's brightness. The block permutation operation does not change pixel values, thus the histograms of encrypted image with/without permutation remain the same. In Fig. 10, we present histograms of the original 'Aerial' image and its corresponding cipher image under *Algorithm-1'*. It can be seen that histogram of the encrypted 'Aerial' image has little statistical similarity to that of the plain image. But the histogram does not show the normal distribution property, which indicates a more secure level, this is because our encryption scheme is based on the 8×8 block unit, the correlation among pixels in the encrypted images cannot be destroyed completely.

For the correlation diagram of images, we initially identify the neighbourhood of diagonal pixels from the original image and the encrypted image. Then 1000 pairs of two adjacent pixels are randomly selected, plot the corre-

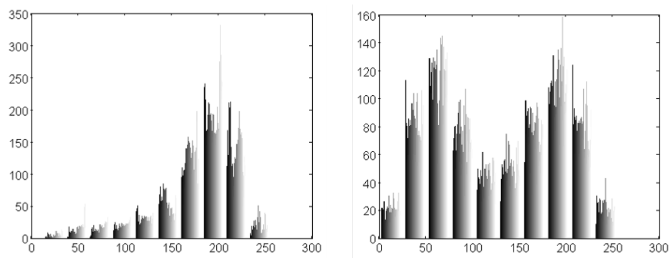


Figure 10: Histogram charts of 'Aerial' image (left: plain image, right: *Algorithm-1'* encrypted image)

lation diagram based on the value of each pixel and its diagonal neighbours. The corresponding diagram of 'Aerial' image encrypted by *Algorithm-1'* is shown in Fig. 11. In the original image, because the correlation between pixels is strong, its correlation chart is shown near linear. After using the algorithm we proposed, the linear property is not shown too much because of the decreased correlation between pixels. However, similarly the linear property cannot be totally removed because we realize encryption at the transformation stage and use 8×8 size block as permutation unit.

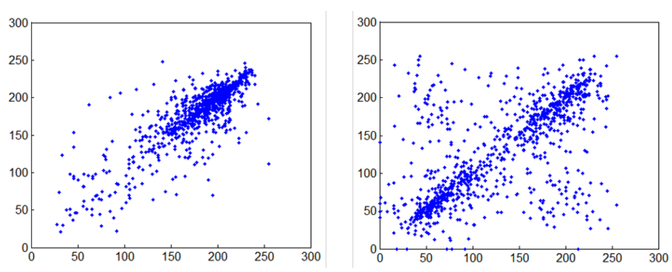


Figure 11: Diagonal correlation charts of 'Aerial' image (left: plain image, right: *Algorithm-1'* encrypted image)

5. Conclusion

In this paper, a new joint image compression and encryption scheme is proposed to realize image encryption at JPEG's transformation stage. We want to emphasize that our proposed encryption technique does not aim at fully confidentiality, because the stage we choose to realize JPEG image encryption and the processing unit (8×8 size block) restrict that some inner properties of the plain image, like high information redundancy, may not be perfectly destroyed.

In our proposed encryption scheme, rather than only using 8×8 DCT for block transformation, we first generate new order-8 orthogonal transforms by introducing an extra rotation angle of π to different stages' butterflies of DCT's flow-graph structure. Then these new transform sets are applied alternatively for transformation, according to the predefined secret encryption key, to realize joint image compression and encryption. Extensive experiments have been conducted to show that our encryption scheme can provide a sufficient level of encryption without scarifying JPEG's good compression ability too much. Moreover, by carefully selecting the number of butterflies for angle rotation, we can control the visual quality of the encrypted images. In the security analysis section, we have provided a detailed security analysis to prove the robustness of our encryption algorithms against various common attacks for multimedia encryption techniques.

We believe there are few extensions we can do in our following work. First, now we only consider JPEG image encryption during transformation, its security level is not so strong. Since most of the existing encryption algorithms are realized after this stage, like the entropy-coding stage encryption, it would be therefore particularly interesting to see whether our current encryption scheme can be integrated into the existing ones so as to enlarge the space for encryption by one dimension. Second, currently we realize encryption by using different order-8 transforms or permuting 8×8 blocks, the original correlation existed in these 8×8 blocks cannot be perfectly removed and the diffusion property that a good cryptosystem should offer is not so strong in our current encryption scheme, therefore next we may try to perform transformation using different block size and to do permutation in small unit, perhaps the bit-plane level of image pixels, in order to improve the confusion and diffusion properties of our encryption scheme.

References

[1] D. R. Stinson, Cryptography: theory and practice, CRC press, 2005.

[2] A. Nag, J. P. Singh, S. Khan, S. Biswas, D. Sarkar, P. P. Sarkar, Image encryption using affine transform and xor operation, in: Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011 International Conference on, IEEE, 2011, pp. 309–312.

[3] S. P. Indrakanti, P. Avadhani, Permutation based image encryption technique, International Journal of Computer Applications (0975–8887) Volume.

[4] M. A. Bani Younes, A. Jantan, Image encrytion using block based transformation algorithm.

[5] A. Mitra, Y. S. Rao, S. Prasanna, et al., A new image encryption ap-

proach using combinational permutation techniques, International Journal of Computer Science 1 (2) (2006) 127–131.

[6] A. Massoudi, F. Lefebvre, C. De Vleeschouwer, B. Macq, J.-J. Quisquater, Overview on selective encryption of image and video: challenges and perspectives, EURASIP Journal on Information Security 2008 (2008) 5.

[7] L. Tang, Methods for encrypting and decrypting mpeg video data efficiently, in: Proceedings of the fourth ACM international conference on Multimedia, ACM, 1997, pp. 219–229.

[8] L. Qiao, K. Nahrstedt, M.-C. Tam, Is mpeg encryption by using random list instead of zigzag order secure?, in: Consumer Electronics, 1997. ISCE'97., Proceedings of 1997 IEEE International Symposium on, IEEE, 1997, pp. 226–229.

[9] C.-P. Wu, C. J. Kuo, Design of integrated multimedia compression and encryption systems, Multimedia, IEEE Transactions on 7 (5) (2005) 828–839.

[10] C.-P. Wu, C.-C. J. Kuo, Fast encryption methods for audiovisual data confidentiality, in: Information Technologies 2000, International Society for Optics and Photonics, 2001, pp. 284–295.

[11] C.-P. Wu, C.-C. J. Kuo, Efficient multimedia encryption via entropy codec design, in: Photonics West 2001-Electronic Imaging, International Society for Optics and Photonics, 2001, pp. 128–138.

[12] M. Van Droogenbroeck, R. Benedett, Techniques for a selective encryption of uncompressed and compressed images, in: Advanced Concepts for Intelligent Vision Systems (ACIVS), 2002.

[13] B. Zeng, S.-K. A. Yeung, S. Zhu, M. Gabbouj, Perceptual encryption of h. 264 videos: Embedding sign-flips into the integer-based transforms, Information Forensics and Security, IEEE Transactions on 9 (2) (2014) 309–320.

[14] S.-K. A. Yeung, S. Zhu, B. Zeng, Perceptual video encryption using multiple 8× 8 transforms in h. 264 and mpeg-4, in: Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on, IEEE, 2011, pp. 2436–2439.

[15] S.-K. A. Yeung, S. Zhu, B. Zeng, Design of new unitary transforms for perceptual video encryption, Circuits and Systems for Video Technology, IEEE Transactions on 21 (9) (2011) 1341–1345.

[16] S.-K. A. Yeung, S. Zhu, B. Zeng, Partial video encryption based on alternating transforms, Signal Processing Letters, IEEE 16 (10) (2009) 893–896.

[17] W.-H. Chen, C. Harrison, S. Fralick, A fast computational algorithm for the discrete cosine transform, communications, IEEE Transactions on 25 (1977) 1004–1011.

[18] K. Kaukonen, R. Thayer, A stream cipher encryption algorithm arcfour (1999).

[19] R. A. Fisher, F. Yates, Statistical tables for biological, agricultural and medical research, Longman, 1938.

[20] A. A. Bruen, M. A. Forcinito, A. G. Konheim, C. Cobb, A. Young, M. Yung, D. Hook, Applied cryptography: protocols, algorithms, and source code in c.

[21] N. Taneja, B. Raman, I. Gupta, Combinational domain encryption for still visual data, Multimedia tools and applications 59 (3) (2012) 775–793.

[22] S. Lian, Multimedia content encryption: techniques and applications, CRC press, 2008.

[23] S. Bahrami, M. Naderi, Encryption of multimedia content in partial encryption scheme of dct transform coefficients using a lightweight stream algorithm, Optik-International Journal for Light and Electron Optics 124 (18) (2013) 3693–3700.

[24] S. S. Agaian, R. G. Rudraraju, R. C. Cherukuri, Logical transform based encryption for multimedia systems, in: Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on, IEEE, 2010, pp. 1953–1957.

[25] W. Li, N. Yu, A robust chaos-based image encryption scheme, in: Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on, IEEE, 2009, pp. 1034–1037.

[26] G. Chen, Y. Mao, C. K. Chui, A symmetric image encryption scheme

based on 3d chaotic cat maps, Chaos, Solitons & Fractals 21 (3) (2004) 749–761.