A Content-Adaptive Joint Image Compression and Encryption Scheme

Peiya Li and Kwok-Tung Lo

Abstract—**For joint image compression and encryption schemes, the encryption power and compression efficiency are commonly two contradictory things. In this paper, we propose a new joint image compression and encryption scheme based on lossy JPEG standard, which aims at encryption power's enhancement, on the premise of maintaining JPEG's compression efficiency. The proposed scheme is image-content-adaptive, since the secret encryption key is generated from the plain-image using BLAKE2 hash algorithm. Three encryption operations are contained in our scheme, alternating new orthogonal transforms transformation, DC coefficients encryption, and AC coefficients encryption. To save the cost for transmitting different encryption keys each time to decoder for decryption when plain-image changes, we propose to embed the encryption key into the entropy encoded bitstream of some AC coefficients, and the whole embedding procedure is controlled by another secret key called embedding key. Extensive experiments are conducted to show that our encryption scheme is JPEG friendly, has good confusion and diffusion properties. Detailed security analysis is also given to illustrate the proposed scheme's persistence to various cryptanalysis strategies.**

Index Terms—**Image encryption, orthogonal transforms, data embedding, security analysis.**

I. Introduction

In recent years, thanks to the rapid growth of computer networks and information technology, a huge amount of multimedia data has been exchanged over various types of social network platforms, such as Facebook and Instagram. In major parts of these exchanged data (image, video or audio), no matter whether they are confidential or private, they need security mechanisms to offer various degree of protection. There are three different ways to protect digital data from eavesdropping and intercepting: cryptography, watermarking and steganography [1], among which cryptography is one of the major techniques for protection. Since images play a crucial role in communication, and raw videos can also be seen as the construction of a sequence of still images (frames), the encryption of image data is receiving more and more attention.

Since the 1970s, a large number of cryptographic techniques have been developed, some of which have been widely adopted as the standardized encryption algorithms, such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES) [2]. When using these conventional algorithms for image encryption, however, the computational cost will be high because of large size of image data. Moreover, after DES/AES, the output cipher-image often appears random, hence little space can be exploited for compression purpose. To accelerate the encryption speed and simplify cryptographic operations, various permutation-only image encryption algorithms had been proposed [3]–[7]. This type of encryption scheme can be easily implemented and is available in both spatial and frequency domain, but it still has some inherent limitations. For example, the permutation-only method cannot change the frequency distribution of pixels because pixel values are not modified, thus it is vulnerable to statistical attack. Additionally, when the size of plain-image is small, the number of permutation arrangements for pixels/blocks will be less than the key space, guessing the permutation mapping using chosen-plaintext attack may break the cryptosystem [8]–[10]. Since most of the images we see on Internet are compressed, the focus of image protection research shifts to integrating image compression procedure with encryption, for the purpose of decreasing encryption/decryption time in image processing and communication.

JPEG and JPEG2000, are two image compression standards created by the Joint Photographic Experts Group committee, in which JPEG2000 was proposed in 2000, later than JPEG, with the intention of superseding their original DCT-based JPEG standard. Secure JPEG2000 (JPSEC), as the security part of the JPEG2000 standard, it provides a framework for security solutions for JPEG2000 images. JPSEC is designed to be an open framework so that new tools can be incorporated in the future. Nonetheless, as of 2017, there are very few digital cameras that encode photos in the JPEG 2000 format, many applications for viewing and editing photos still do not support it. Therefore, our security mechanism's design mainly focuses on the lossy JPEG standard. In Fig. 1, the baseline encoding procedure of JPEG is given, from which we can observe that it mainly contains four stages: DCT transformation, quantization, zigzag scan, and entropy coding. For JPEG decoding process, stages contained are just the reverse of the four stages in encoding procedure.

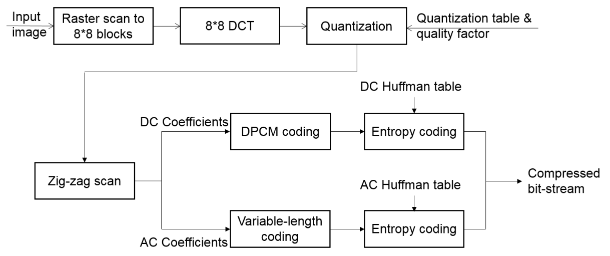


Fig. 1: Block diagram of JPEG encoder.

JPEG, as a common image compression standard applied

The authors are with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China (e-mail: yolanda.peiya@connect.polyu.hk; kwok.tung.lo@polyu.edu.hk).

for image storage and transmission, it greatly saves the storage space and transmission bandwidth. Hence, study on joint JPEG image compression and encryption has important theoretical and practical values, and it has attracted many researchers' attention. Existing joint JPEG image compression and encryption schemes are mainly realized through introducing encryption techniques into one stage or several stages of JPEG standard. In [11], Au-Yeung *et al.* proposed to realize perceptual video encryption in the transformation stage of H.264 and MPEG-4 using multiple 8×8 transforms, which can be also adopted for JPEG image encryption. Three different encryption algorithms with different transforms had been proposed, and *Algorithm-3* had been proved to have the best balance between encryption and compression performance. Their scheme could achieve controllable degree of encryption without endangering the compression performance of the underlying video compression standard, but the 8×8 blocks' independently encryption technique possessed little diffusion property, which may be vulnerable to differential attack. In [12], [13], they proposed methods for encryption at JPEG's quantization stage, and this stage's encryption techniques were often accompanied with other stages' encryption to offer a higher protection power. Later, Qian *et al.* in [14] pointed out that algorithm in [12] was not secure enough, because when an adversary used all Huffman codes and set all appended bits to zero, he/she could reconstruct a contour of the plain-image. To improve the protection power, [14] proposed to first randomly select only part of JPEG's bitstream segments, then use a stream cipher to encrypt all appended bits of the DC and AC Huffman codes inside these selected segments, and finally obtain a smaller sized JPEG cipher-image. Thus, with limited Huffman codes, adversary cannot recovery the contour of the plain-image. For encryption realized at JPEG's entropy coding stage, Wu *et al.* [15] achieved JPEG image encryption by using different Huffman tables for different input symbols. This scheme could obtain very high visual degradation without sacrificing the compression performance, but it was not format compliant to JPEG standard, since the decoder needed to know the Huffman tables used for encoding in order to decompress the encrypted bitstream correctly [16]. Additionally, [17] and [18] had pointed out that [15]'s encryption scheme was vulnerable to chosen-plain-text/known-plain-text attacks.

In [19], Zhang *et al.* pointed out that some joint image compression and encryption cryptosystems existed security flaws, like methods in [20], [21]. They said that the secret keys controlled interval allocation encryption operations were equivalent to confusion without diffusion, which resulted in some loopholes for some classic attacks, like chosen-plaintext/known-plaintext attacks. To overcome this defect, they argued that the encryption scheme should not only rely on the confusion-diffusion architecture but also should have varying encryption parameters related to plaintext.

In our work, we propose a joint image compression and encryption scheme which has good diffusion property and fine compression efficiency. The whole cryptosystem is based on lossy JPEG standard, and is plain-image-content-adaptive, since the secret key we use for encryption is dependent on the plain-image. To generate the plain-image sensitive key, we use BLAKE2 hashing algorithm [22] taking plain-image as input, and outputs a 256-bit random hash value, denoted as Key_1 and named encryption key, which is used to control the following encryption operations. BLAKE2 is very sensitive to changes occurred in the input, hence different input plain-images result in different Key_1 values, and these values must be available to the decoder for cipher-images' recovery. In order to save the cost for transmitting different Key_1 to decoder each time, we embed Key_1 into the entropy encoded bitstream of certain AC coefficients through some embedding rules, and the embedding positions are determined by another 256-bit secret key Key_2, named embedding key, which is previously known and only known to the encoder and decoder. The secret Key_2 could remain unchanged for a period of time, decided by encoder and decoder, even though the plain-images for compression and encryption change. Our encryption operations include new orthogonal transforms transformation, DC coefficients encryption and AC coefficients encryption. Experiment results have shown that the whole encryption scheme is efficient and has good compression and security features.

The rest of the paper is organized as follows. Section II explains the implementation details of our encryption operations, and the corresponding encryption and decryption algorithms are also given. Experimental results are presented in Section III. Section IV analyses the cryptosystem's security level against various types of attacks. Section V presents concluding remarks and our future research directions.

II. Implementation of Encryption Operations

In this section, we introduce the realization details of our encryption operations, which include new orthogonal transforms transformation, DC encryption by block permutation and XOR operation, and AC encryption through data embedding. Their design principles are to achieve three objectives: (a) suppression of bitstream size increment, (b) format-compliant encryption, and (c) confusion and diffusion properties' enhancement. Section II-A explains how we generate different new orthogonal transforms, and compare their coding efficiency with other transforms. Methods for encrypting DC coefficients are given in Section II-B, and Section II-C explains our data embedding/extracting strategy for encrypting AC coefficients, which is controlled by the predefined embedding key Key_2. The encryption algorithm and decryption algorithm are presented in Section II-D.

Let a given plain-image be of size $H\times W$, where *H* and *W* represent the height and width of the plain-image in pixels. We take it as the input of BLAKE2 algorithm, and produce a 256-bit random value Key_1 to control encryption/decryption operations. Another 256-bit secret parameter Key_2 is predefined, and is only known to the encoder and decoder. For generating pseudo-random key-streams from Key_1 and Key_2, we also use BLAKE2 algorithm taking Key_1 and Key_2 as the initial seed key, computation formula is given as following

$$K_{n+1} = H(K_n)(n = 0, 1, 2, ...) \tag{1}$$

where $H(\cdot)$ is the BLAKE2 hash function, K_0 is Key_1 or Key_2. We denote these two pseudo-random key-streams as K_{enc-1} and K_{enc-2}.

A. New orthogonal transforms transformation

In JPEG, 8×8 type-II DCT is used for blocks' transformation because it can provide an efficient compaction performance with separability property when high correlation is existed among inter-pixels [23] - which is a general phenomenon in most natural images. In our work, we modify JPEG's transformation stage by allowing more orthogonal transforms for 8×8 blocks' transformation, rather than only DCT. To maintain the excellent compression performance of JPEG, coding efficiency of our newly generated transforms should be similar to DCT or just drop slightly.

In [24], a fast computational algorithm for DCT-II was developed in the form of matrix decomposition, and was illustrated by a flow-graph structure. In general, an order-*N* DCT-II matrix can be represented by the following recursive form:

$$\left[\ C_N^{II}\ \right] = \left[\ P_N\ \right] \begin{bmatrix} P_{N/2}^t C_{N/2}^{II} & 0 \\ 0 & R_{N/2} \end{bmatrix} \left[\ B_N\ \right], \tag{2}$$

$$\left[\ B_N\ \right] = \frac{\sqrt{2}}{2} \begin{bmatrix} I_{N/2} & \bar{I}_{N/2} \\ \bar{I}_{N/2} & -I_{N/2} \end{bmatrix},$$

where $[P_N]$ is a $N\times N$ 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.

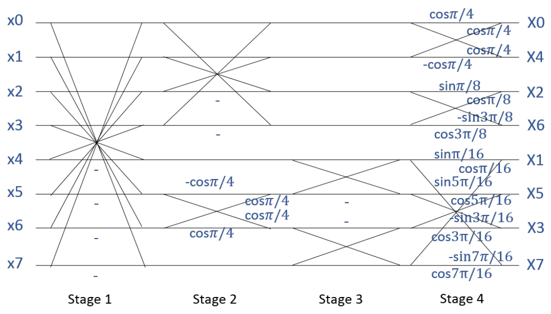


Fig. 2: Flow graph of 8-point (1-D) DCT.

In Fig. 2, the 8×8 DCT-II's flow-graph is shown. When $N = 8$, Eqn. (2) becomes

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

$$\left[\ C_4^{II}\ \right] = \left[\ P_4\ \right] \begin{bmatrix} P_2^t C_2^{II} & 0 \\ 0 & R_2 \end{bmatrix} \left[\ B_4\ \right],$$

$$\left[\ P_4\ \right] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\left[\ P_2\ \right] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \left[\ C_2^{II}\ \right] = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

$$\left[\ R_2\ \right] = \begin{bmatrix} \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \\ -\sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} \end{bmatrix},$$

Computation of $[R_2]$ is equal to the second butterfly (from top to bottom) of Stage-4 in Fig. 2.

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

$$\left[\ R_4\ \right] = \left[\ M1\ \right] \left[\ M2\ \right] \left[\ M3\ \right], \text{where}$$

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

Computation of $[M1]$ is equal to the two butterflies denoted in the bottom half of Stage-4 in Fig. 2.

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

Computation of $[M2]$ is equal to the two butterflies denoted in the bottom half of Stage-3 in Fig. 2.

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

Computation of $[M3]$ is equal to the bottom half part of Stage-2 in Fig. 2.

To generate new orthogonal transforms, we introduce different rotation angles into the four butterflies at Stage-4, which can be represented by modifying Eqn. (3) using the following formula:

$$\left[\ C_8^{II}\ \right] = \left[\ P_8\ \right] \left[\ T_1\ \right] \begin{bmatrix} P_4^t C_4^{II} & 0 \\ 0 & R_4 \end{bmatrix} \left[\ B_8\ \right], \tag{4}$$

where $[T_1] = diag(\cos\alpha_1, \cos\alpha_1, \cos\alpha_2, \cos\alpha_2, \cos\alpha_3, \cos\alpha_4, \cos\alpha_4, \cos\alpha_3)$, α_1 = 0 or π, $\alpha_i \in$ (0, $\frac{\pi}{3}$, $\frac{2\pi}{3}$, π) (i = 2, 3, or 4), and the normalized coefficients are ignored here. The reason for allowing α_1 to have only two rotation angles' choice (0 or π) is because α_1 will influence the DC coefficient, changing it too much may have a great impact on the overall coding efficiency of the new transform. Using Eqn. (4), we can totally produce 128 different orthogonal transforms, and we name the new transform set as *TS1*.

To evaluate the coding efficiency of *TS1*, we compare its performance with DCT, transforms used in [11]'s *Algorithm-3*, and transforms used in [25] through a statistical model. For the *Algorithm-3* in [11], they used different transform sets for column's transformation and row's transformation. For the row's transformation, 16 orthogonal transforms were generated by introducing two rotation angles (0 or π) into Stage-4's four butterflies of Fig. 2, and all eight rows used the same transform matrix for transformation, decided by the secret key. For the column's transformation, totally 256 orthogonal transforms were generated by introducing rotation angles (0 or π) into eight butterflies of Fig. 2 (2 butterflies in the top half of Stage-2, 2 butterflies in the bottom half of Stage-3, and 4 butterflies in Stage-4), and each column used different transform matrices, also determined by the secret key.

The statistical model we use for comparison is the one-dimensional, zero-mean, unit-variance first-order Markov process with adjacent element correlation ρ. For a n-dimensional input vector X sampled from this model, the (i, j)th element of the covariance matrix $[C_X]$ is equal to $\rho^{|i-j|}$. The efficiency of the transform $[T]$ is defined on the transform domain covariance matrix $[C_Y]$ of vector Y, where

$$\begin{aligned} Y &= [T]\,X \\ [C_Y] &= E[Y \cdot Y^t] \\ &= [T][C_X][T]^t \\ &= \begin{bmatrix} s_{11} & \cdots & \cdots & s_{1n} \\ \vdots & & & \vdots \\ s_{n1} & \cdots & \cdots & s_{nn} \end{bmatrix} \end{aligned}$$

We use the criteria of transform coding gain [26] and transform efficiency [27] for performance comparison. Transform coding gain (G_{TC}) measures the energy compaction ability of the transform in transform coding system, while transform efficiency (η) measures the ability of an unitary transform to decorrelate the input vector X, computing formulas of these two criteria are given as follows:

$$G_{TC} = \frac{\frac{1}{n}\sum_{i=0}^{n-1} s_{ii}}{\left(\prod_{i=0}^{n-1} s_{ii}\right)^{1/n}}$$

$$\text{Efficiency } \eta = \frac{\sum_{i=0}^{n-1} |s_{ii}|}{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} |s_{ij}|} \times 100\%$$

The average transform coding gain G_{TC} and average transform efficiency η versus different correlation coefficients ρ for DCT, transforms in *TS1*, transforms in [11]'s *Algorithm-3*, and transforms in [25] are shown in Table I and Table II respectively. It is clear that the coding efficiency of our newly generated transform set is better than that in [11] and [25], and very close to DCT[1].

When using *TS1* for transformation in each 8×8 block, we transform all eight rows using the same transform matrix, and transform eight columns using different transform matrices, all used transform matrices are determined by K_{enc-1}. The reason why one transform is selected for each 8×8 block's eight rows is because using different transforms in the first dimension will generate misalignment in the second dimension, relatively larger high-frequency coefficients will be produced, and this causes larger quality drop in the decrypted cipher-image.

[1]In our experiment, we use 32 digits precision to display numbers, and the minor differences in G_{TC} and η values of *TS1* and DCT can be seen.

TABLE I: Transform coding gain for different order-8 transforms

ρ	0.80	0.85	0.90	0.95
DCT	2.4162	3.0386	4.2424	7.6312
TS1	2.4162	3.0386	4.2424	7.6312
Ref [11]	2.2092	2.7553	3.8160	6.8119
Ref [25]	2.3065	2.8852	4.0064	7.1666

TABLE II: Transform efficiency for different order-8 transforms

ρ	0.80	0.85	0.90	0.95
DCT	0.8497	0.8695	0.8984	0.9399
TS1	0.8497	0.8695	0.8984	0.9399
Ref [11]	0.7748	0.8026	0.8440	0.9056
Ref [25]	0.8105	0.8344	0.8698	0.9219

B. DC coefficients encryption

To make the encrypted image become more disorder, after JPEG quantization process, we propose to permute all order-8 quantized blocks, and the permutation vector is generated from Fisher-Yates shuffle [28] controlled by K_{enc-1}.

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 blocks, the random integer in each loop is obtained from K_{enc-1}, which can be described as following:

Random Integer Generation

1) Chose an integer r satisfying $2^r \geq n$;
2) Obtain r bits from K_{enc-1} 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;

After performing the block permutation operation using S for all 8×8 quantized blocks, we change DC coefficient in each 8×8 permuted block using Eqn. (5), to further improve the diffusion and confusion properties.

$$dc_i = dc_i \oplus dc_{i-1} \oplus dc_{i-2} \oplus \cdots \oplus dc_1, \tag{5}$$

where dc_i is the DC coefficient of the i^{th} 8×8 permuted block, $i = 1, 2, \cdots, (H \times W)/64$. For recovering these confused DC coefficients, Eqn. (6) can be used:

$$dc_j = dc_j \oplus dc_{j-1}, \tag{6}$$

where j starts from $(H \times W)/64$, and ends with 2. The DC coefficients confusion operation could increase the cryptosystem's resistance ability against differential attack, but it may cause irregular lines/stripes in the encrypted images.

C. AC coefficients encryption

To save the cost for transmitting different encryption key Key_1 when plain-image changes, we embed the 256-bit Key_1 into non-zero AC coefficients with run-length being 0. The reason for choosing this kind of non-zero AC coefficients is to control the final bitstream size, since more number of zeros existed before a non-zero AC coefficients mean a longer Huffman code to represent this coefficient.

In order to ensure the embedded Key_1 is unavailable to attackers, we use K_{enc-2} to decide embedding positions for these 256-bit data. In our data embedding strategy, each qualified non-zero AC coefficient carries one secret bit. Suppose after variable-length codes (VLC) coding for all AC coefficients, the number of qualified non-zero AC coefficients is L ($L > 256$), we use K_{enc-2} controlled Fisher-Yates shuffle to permute these L coefficients' positions, then save the first 256 permuted coefficients' positions into an array $Index$ for the control of data embedding. The whole operation is described in Fig. 3.

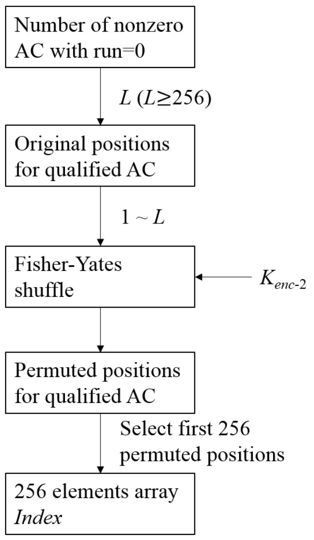

Number of nonzero AC with run=0
L (L≥256)
Original positions for qualified AC
1 ~ L
Fisher-Yates shuffle
K_enc-2
Permuted positions for qualified AC
Select first 256 permuted positions
256 elements array Index

Fig. 3: Positions selection for data embedding.

For data embedding strategy, when one nonzero AC coefficient with run being zero appears, we first check whether this coefficient's position is in array $Index$ or not. If not, do not change the Huffman coding result of this coefficient. If yes, increase the coefficient's category by 1, then look up JPEG's Huffman table to get its corresponding Huffman code, concatenate the changed Huffman code with the appended bits of this AC coefficient to generate new entropy encoded bits Q. Then append one bit data of Key_1 to the final position of Q and produce Q^*. Q^* is the final compressed and encrypted bitstream data that will be transmitted to the decoder for decryption and decompression. One example is given in Fig. 4 to better explain our data embedding strategy.

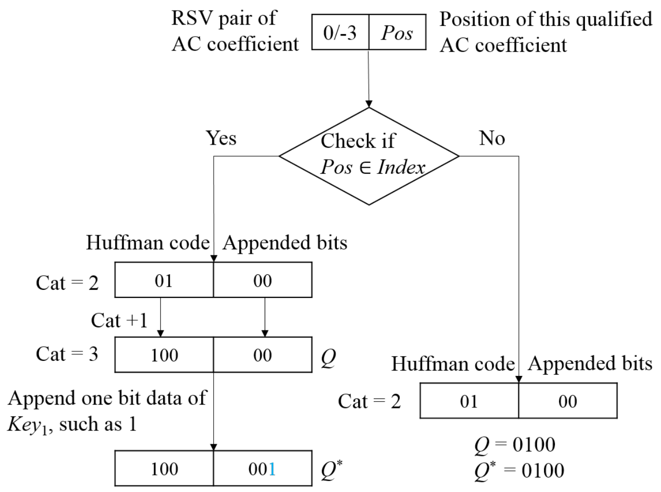


Fig. 4: Example of data embedding strategy.

For extracting the embedded 256-bit Key_1 from Q^* correctly, the decoder needs to know Key_2 to produce K_{enc-2} and to do Fig. 3's procedures, and obtain the 256 embedding positions array $Index$. Given the entropy encoded bits of one AC coefficient, decoder first checks if its run-length is 0 through searching the Huffman table of AC coefficients. If the run-length is 0, and position for this qualified AC coefficient is in array $Index$, then obtain the appended bits through the Huffman table denoted category for this non-zero AC coefficient, and the final bit in appended bits is the one bit data of Key_1. One example of data extracting is presented in Fig. 5, which is the reverse procedure of Fig. 4.

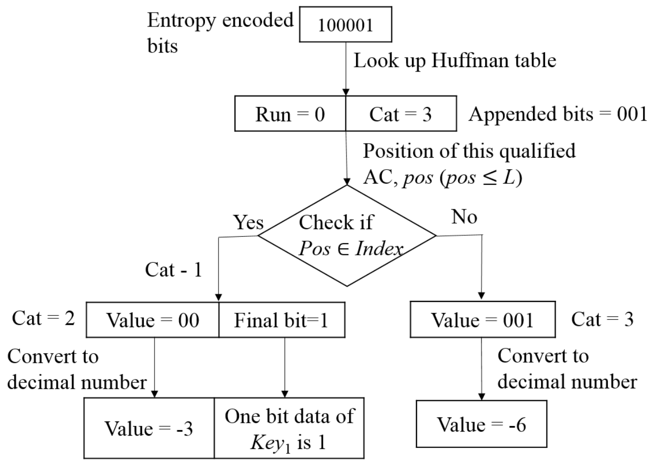


Fig. 5: Example of data extracting strategy.

D. Encryption and Decryption Algorithms

The realization procedures of our proposed cryptosystem are presented in Fig. 6, and the encryption and decryption algorithms can be described as follows:

Encryption Algorithm

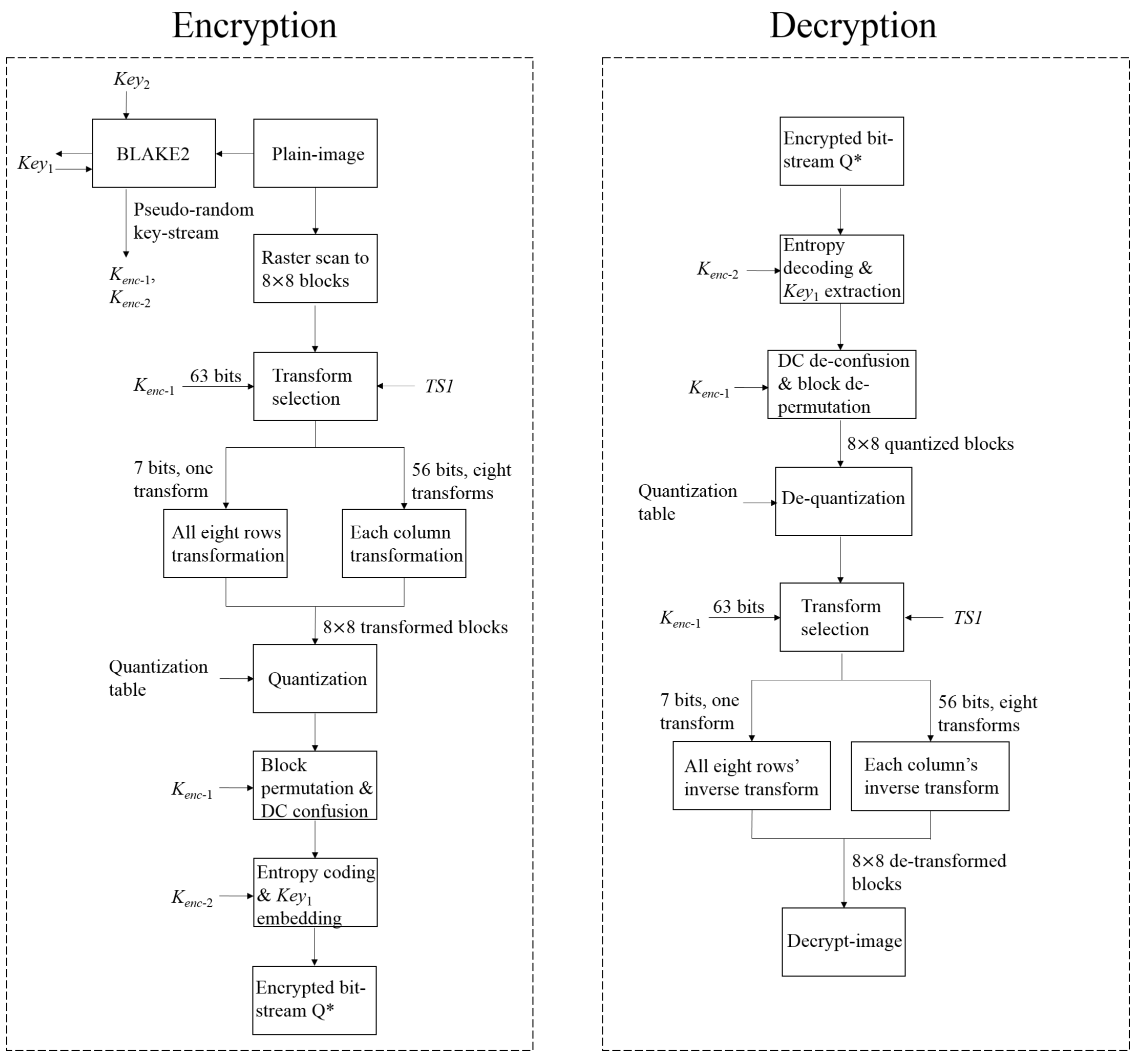

Encryption
Decryption
Key2
BLAKE2
Plain-image
Key1
Pseudo-random key-stream
Kenc-1, Kenc-2
Raster scan to 8×8 blocks
Kenc-1
63 bits
Transform selection
TS1
7 bits, one transform
56 bits, eight transforms
All eight rows transformation
Each column transformation
8×8 transformed blocks
Quantization table
Quantization
Block permutation & DC confusion
Kenc-2
Entropy coding & Key1 embedding
Encrypted bit-stream Q*
Entropy decoding & Key1 extraction
DC de-confusion & block de-permutation
8×8 quantized blocks
De-quantization
All eight rows' inverse transform
Each column's inverse transform
8×8 de-transformed blocks
Decrypt-image

Fig. 6: Encryption and decryption procedures of our scheme.

Step-1: Take plain-image as input of BLAKE2 algorithm to generate encryption key Key_1, and predefine embedding key Key_2;

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

Step-2.1: Get 63 bits from Key_1 produced pseudo-random key-stream K_{enc-1};

Step-2.2: Use the first 7 bits to select one transform from *TS1* for *all* rows' transformation in the 1^{st} dimension;

Step-2.3: Use the following 7×8 bits to select one transform from *TS1* for *each* column's transformation in the 2^{nd} dimension;

Step-3: Repeat *Step-2* until all 8×8 blocks are transformed, quantize all these transformed blocks using JPEG's quantization table;

Step-4: Use Fisher-Yates shuffle and K_{enc-1} to perform 8×8 blocks' permutation, and do DC coefficients confusion using Eqn. (5);

Step-5: Perform JPEG's entropy encoding procedure for all processed 8×8 blocks, encrypt AC coefficients with 256-bit Key_1's embedding, controlled by K_{enc-2}, and generate the final encrypted bitstream Q^*.

Decryption Algorithm

Step-1: Perform entropy decoding procedure of JPEG, using Q^* and K_{enc-2} to decode DC/AC coefficients and extract Key_1;

Step-2: Use Eqn. (6) to recover the encrypted DC coefficients, and put the permuted 8×8 blocks back to their original positions, realized by Key_1 produced pseudo-random key-stream K_{enc-1} and Fisher-Yates shuffle;

Step-3: De-quantize all 8×8 blocks using JPEG's order-8 quantization table;

Step-4: For each 8×8 block obtained in *Step-3*, do

Step-4.1: Get 63 bits from K_{enc-1};

Step-4.2: Use the first 7 bits to select one transform from *TS1* for *all* rows' inverse-transformation in the 1^{st} dimension;

Step-4.3: Use the following 7×8 bits to select one transform from *TS1* for *each* column's inverse-transformation in the 2^{nd} dimension;

Step-5: Repeat *Step-4* until all 8×8 blocks are de-transformed,

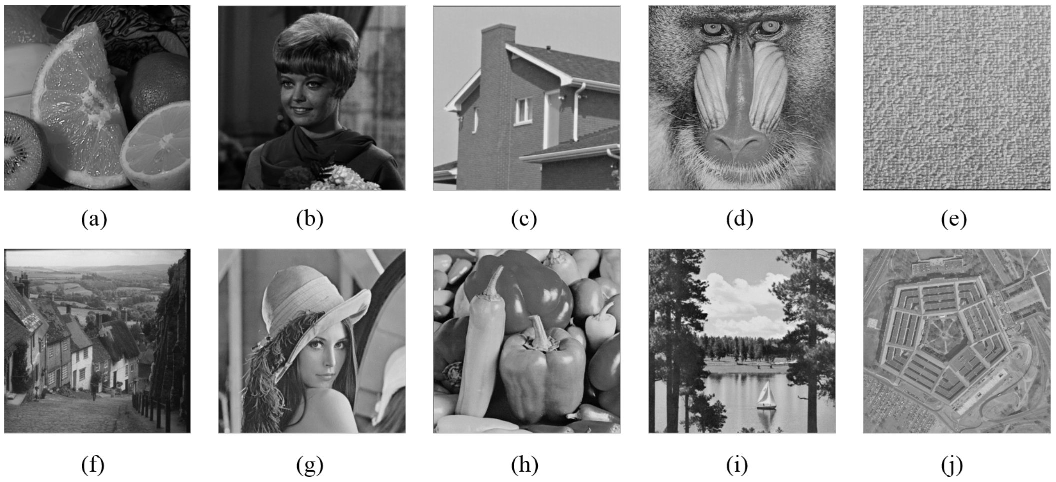


Fig. 7: Ten test images. (a) Fruits 256×256. (b) Girl 256×256. (c) House 256×256. (d) Baboon 512×512. (e) Raffia 512×512. (f) Goldhill 512×512. (g) Lena 512×512. (h) Peppers 512×512. (i) Sailboat 512×512. (j) Pentagon 1024×1024.

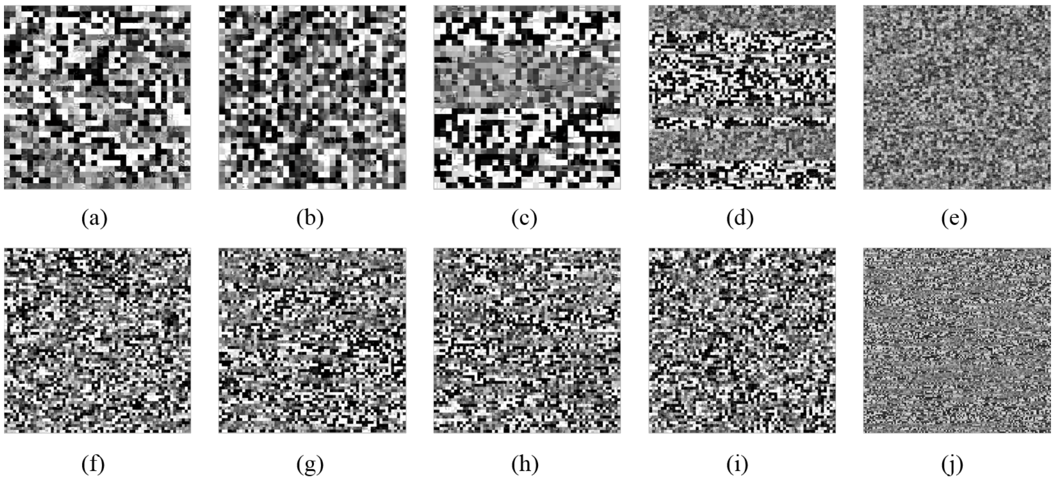


Fig. 8: Ten cipher-images encrypted by the proposed scheme.

then we can obtain the decrypted image.

III. Experimental Results

In this section, the perceptual security of our proposed scheme and the compression performance on the encrypted images are evaluated experimentally. Ten images with different sizes are used in our experiment, shown in Fig. 7. Perceptual security refers to the perceptual distortion of cipher-image with respect to the plain-image, and the peak signal-to-noise ratio (PSNR) criterion is adopted to measure it. We encrypt these ten images using our proposed schemes with QF being 20, and their corresponding cipher-images are shown in Fig. 8. It can be observed that these cipher-images are disorder and no outline information about their plain-images can be seen.

For the compression performance, it is influenced by the quality factor (QF), and higher QF value results in less compression and larger bitstream size. Here, we use two criteria, the bit per pixel (BPP) value and the PSNR value of decrypted image, to evaluate the compression performance of our proposed scheme. The BPP value is obtained from dividing the bitstream size with image size. The two values for various images under different QF values (QF = 20, QF = 40, QF = 60, and QF = 80) are presented in Table III through VI, and they are compared with the results of JPEG. From the four tables, we can observe that the BPP values of various cipher-images resulted by our proposed method under different QF values are a little larger than results of JPEG, which confirms that our proposed encryption scheme is compression friendly to JPEG.

To better illustrate our scheme's performance, we use four images ('Fruits', 'Baboon', 'Raffia', and 'Pentagon') for encryption and decryption, when QF ranges from 10 to 90. Their BPP-PSNR curves and PSNR-compression ratio curves are plotted in Fig. 9 and Fig. 10, and they are compared with JPEG, [11]'s *Algorithm-3* and the encryption scheme in [25]. From the BPP-PSNR curves in Fig. 9, we can see that when the encryption key is not known and only IDCT is used for different cryptosystems' decompression and decryption, four images' PSNR values of our encryption scheme have

TABLE III: Comparison of compression performance when QF = 20

Image	Proposed		JPEG	
	BPP	PSNR	BPP	PSNR
Fruits	0.5018	28.9958	0.4546	28.9957
Girl	0.4213	32.8159	0.3647	32.8161
House	0.4098	32.6707	0.3528	32.6707
Baboon	0.7778	25.2637	0.7440	25.2637
Raffia	0.7873	28.8030	0.7590	28.8030
Goldhill	0.4869	30.8849	0.4381	30.8849
Lena	0.3911	32.9625	0.3517	32.9624
Peppers	0.3923	32.4363	0.3492	32.4362
Sailboat	0.5418	30.0710	0.4931	30.0712
Pentagon	0.4933	29.1710	0.4591	29.1710

TABLE IV: Comparison of compression performance when QF = 40

Image	Proposed		JPEG	
	BPP	PSNR	BPP	PSNR
Fruits	0.8234	30.9837	0.7654	30.9837
Girl	0.6234	35.0051	0.5544	35.0026
House	0.6125	34.9489	0.5445	34.9492
Baboon	1.2290	27.3780	1.1883	27.3779
Raffia	1.1754	31.2211	1.1417	31.2211
Goldhill	0.7658	33.1318	0.7070	33.1340
Lena	0.5841	35.1285	0.5378	35.1285
Peppers	0.5903	34.2135	0.5404	34.2139
Sailboat	0.8228	31.9617	0.7647	31.9621
Pentagon	0.7992	30.8560	0.7700	30.8560

TABLE V: Comparison of compression performance when QF = 60

Image	Proposed		JPEG	
	BPP	PSNR	BPP	PSNR
Fruits	1.1081	32.5396	1.0404	32.5389
Girl	0.8161	36.3435	0.7374	36.3416
House	0.8065	36.3645	0.7383	36.3644
Baboon	1.6215	29.1476	1.5832	29.1476
Raffia	1.5042	32.7025	1.4606	32.7026
Goldhill	1.0166	34.5579	0.9472	34.5580
Lena	0.7638	36.4570	0.7199	36.4570
Peppers	0.8011	35.2865	0.7384	35.2865
Sailboat	1.0858	33.0761	1.0152	33.0762
Pentagon	1.0979	32.0463	1.0607	32.0463

TABLE VI: Comparison of compression performance when QF = 80

Image	Proposed		JPEG	
	BPP	PSNR	BPP	PSNR
Fruits	1.6835	35.5861	1.6057	35.5852
Girl	1.2245	38.5559	1.1329	38.5569
House	1.2109	39.0577	1.1290	39.0568
Baboon	2.4414	32.5956	2.3802	32.5955
Raffia	2.1832	35.4463	2.1329	35.4462
Goldhill	1.5321	36.9879	1.4506	36.9896
Lena	1.1884	38.5349	1.1365	38.5357
Peppers	1.2524	36.7890	1.1813	36.7894
Sailboat	1.6495	34.9564	1.5714	34.9560
Pentagon	1.7358	34.3140	1.6898	34.3139

the largest drop, which means a higher perceptual security. Additionally, when the key is available to the decoder, the PSNR values obtained by our scheme are the most close to JPEG, which indicates the coding efficiency of our new transform set is better than that in [11] and [25], matching the results of coding efficiency comparison using the statistical model in Section II-A. For the compression ratio values listed in Fig. 10, it can be seen that when PSNR is fixed, our encryption scheme possesses the highest compression ratio compared with [11] and [25]. In Table VII, we also compare the encryption efficiency for different sizes of plain-images under our proposed scheme[2], DES, and only JPEG compression process. The simulation environment is MATLAB R2014a in 64 bit operating system, 3.50 Ghz, 16 GB RAM, Intel Core i7-4770K. QF is set to be 20.

TABLE VII: Efficiency comparison for different encryption schemes

Image size	Encryption time of our scheme (s)	Encryption time of DES (s)	JPEG compression (s)
256×256	0.98	34.34	0.63
512×512	2.64	226.15	1.44
1024×1024	8.33	2604.09	3.61

IV. Security Analysis

In the previous section, we have proved that our proposed encryption scheme can achieve a good compression ratio close to JPEG, and it is format compliant to JPEG. In this section, we analyse the encryption performance by evaluating its robustness against various cryptanalysis techniques, such as ciphertext-only attack, differential attack, etc.

A. Ciphertext-only attack

In typical cryptographic attacking methods, ciphertext-only attack is the most basic and realistic one in which only the encrypted data is available to attackers. And one of the common techniques for ciphertext-only attack is to try all possible keys in the brute-force manner. To make brute-force attack infeasible, cryptosystem's key space should be large enough. In our encryption scheme, we use 256-bit Key_1 to control all encryption operations, thus obtain a 2^{256} key space, which is not feasible for attackers to guess. However, since we embed the 256-bit Key_1 into 256 nonzero AC coefficients with run-length being zero, attackers may try to guess the 256 embedding positions to recover Key_1. In our embedding strategy, suppose the number of qualified AC coefficients is $L(L > 256)$, we first use K_{enc-2} to permute these coefficients' positions, then select the first 256 positions for data embedding. Computation complexity for guessing these selected positions is $\binom{L}{256}$, which is greater than $\left(\dfrac{L-255}{256}\right)^{256}$. In our scheme, after JPEG quantization process, on average each 8×8 block should have at least one qualified AC coefficient, for an plain-image of size 256×256, totally 1024 8×8 blocks, the minimum value for L would be 1024, $\left(\dfrac{L-255}{256}\right)^{256} \approx 3^{256} > 2^{256}$, which illustrates that breaking our cryptosystem

[2]Computation cost of BLAKE2 has been included.

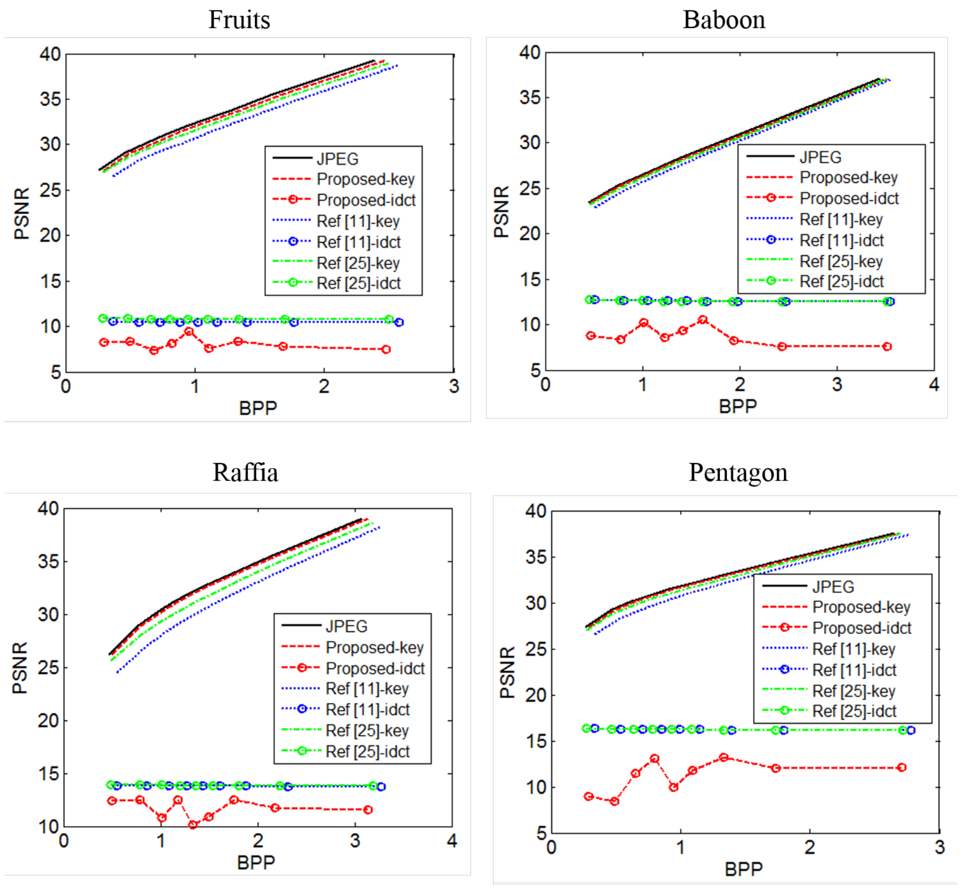


Fig. 9: Comparison of BPP-PSNR curves for different encryption schemes

through guessing the 256 embedding positions is still very hard.

B. Jigsaw puzzle solver attack

In [29], they proposed to regard jigsaw puzzle solvers as one of attack methods on encryption. They thought the safety of the block-scrambling based image encryption schemes might be compromised by jigsaw puzzle solvers. Considering the 8×8 block permutation is contained in our cryptosystem, thus we analyse the proposed scheme's performance against jigsaw puzzle solver attack. Five types of jigsaw puzzles are proposed in [29], and our scheme can be classified into the Type 1 puzzle because only block scrambling operation is done on plain-images. For jigsaw puzzles assembling task, a large amount of jigsaw pieces will increase the assembling complexity and computing time, hence here we only use the three 256×256 images ('Fruits', 'Girl', and 'House') for testing, and the jigsaw piece size is 8×8 pixels. The same three measures (Direct comparison Dc, Neighbor comparison Nc, and Largest component Lc) are used for evaluation, which are ranged in $[0, 1]$, and a larger value means a higher compatibility. In [29], the assembling results for Type 1 puzzles with piece size being 8×8 were not given, thus we first use the only 8×8 blocks scrambling encryption method and our proposed method to encrypt the three images, then apply [29]'s jigsaw puzzle solver to recover these cipher-images, corresponding experimental results are shown in Table VIII. Based on the low values of three measures, we could think that when jigsaw piece size is 8×8, performance of the jigsaw puzzle solver is not so good, therefore our proposed scheme can resist this type of attack.

TABLE VIII: Jigsaw puzzle solver results

Image	Scramble only			Proposed		
	Dc	Nc	Lc	Dc	Nc	Lc
Fruits	0.000	0.464	0.289	0.001	0.000	0.001
Girl	0.001	0.301	0.089	0.000	0.000	0.001
House	0.000	0.168	0.063	0.000	0.000	0.001

C. Replacement attack and sketch attack

Replacement attacks can be divided into two categories: direct replacement attack and correlation-based replacement attack, in which attackers try to recover the plain-image by replacing the encrypted parameter with other ones or those unencrypted image data. In our encryption scheme, we don't rank the information contained in the plain-image, but to encrypt the whole image together, therefore the correlation-based attack is infeasible for attacking our scheme. Here one direct replacement attacking method [15] is analysed by assigning all DC coefficients to 128 and all AC coefficients to positive. We use our proposed scheme, [11]'s *Algorithm-3*, [25]'s encryption scheme to encrypt 'Baboon' image, the

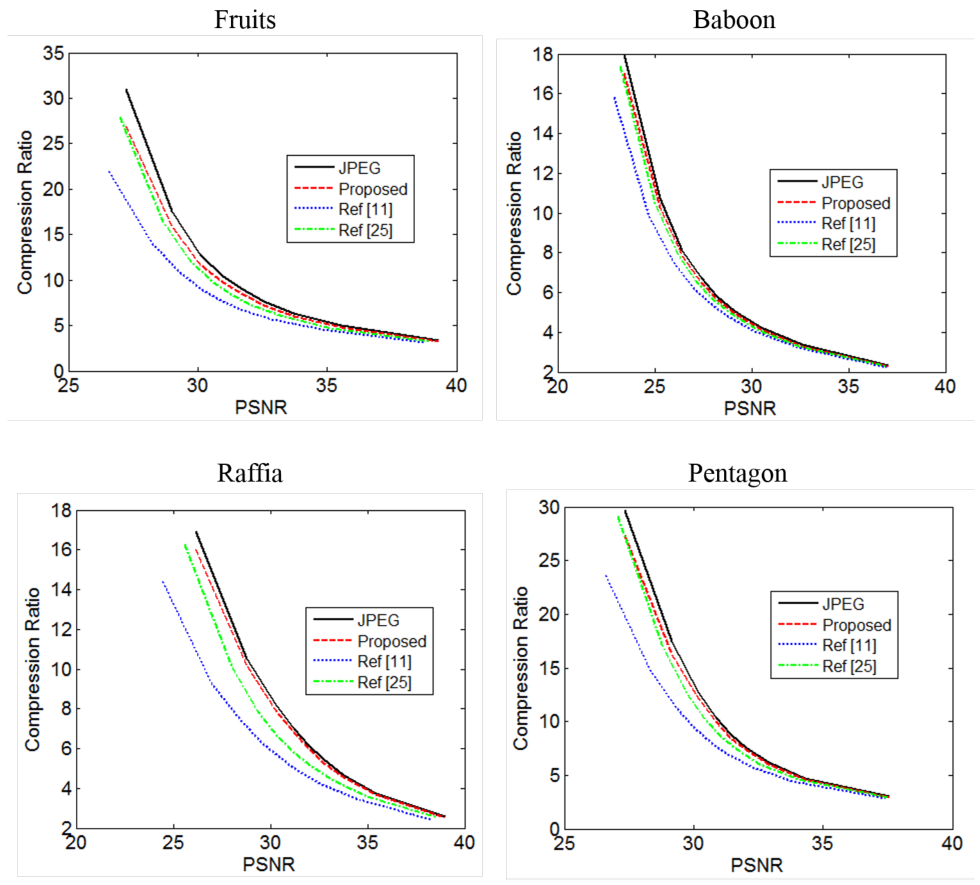


Fig. 10: Comparison of PSNR-compression ratio curves for different encryption schemes

encrypt images and attacking results are shown in Fig. 11. From Fig. 11, we can observe that no information of the plain Baboon image can be distinguished in our scheme, which means a great resistance ability against this replacement attack, while for [11] and [25], they both disclose some contour information of the plain Baboon image.

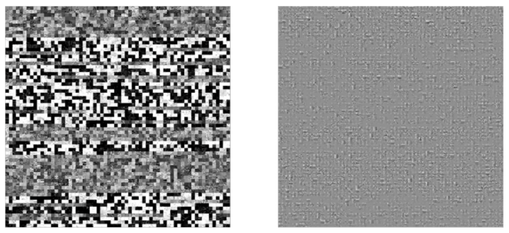
Sketch attack generally tries to sketch the outline of the original image/video frame directly from the encrypted image/video. In [30], several conventional sketch attacks for JPEG images had been introduced, and a new sketch attack based on macroblock bitstream size was proposed for sketching H.264/AVC video frames. Two methods to resist this type of attack were reported, one is to do macroblocks permutation while ensuring format-compliance, another is to diffuse the visual information of a region into other regions. The main disadvantage indicated in [30] when taking these two encryption methods was the reduced compression efficiency. In our proposed cryptosystem, we take these two methods (8×8 blocks' permutation and DC coefficients confusion) while keeping format compliance, and the final compression efficiency of JPEG is not compromised too much, therefore we think our encryption method is robust against sketch attack.

D. Key sensitivity analysis

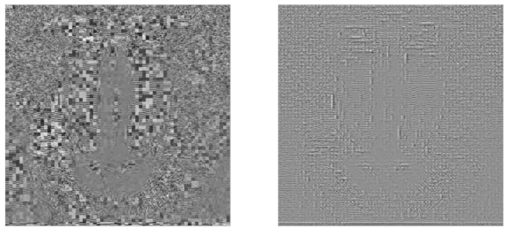
According to Kerckhoff's principle, the security of an encryption scheme should rely on the secrecy of the encryption and decryption keys instead of the encryption algorithm itself [31]. A good cryptosystem should be extremely sensitive with respect to the key used in the algorithm, which should satisfy two conditions to indicate a high key sensitivity level [32]: 1) Completely different cipher-images should be generated when slightly different encryption keys are used to encrypt the same plain-image; 2) The cipher-image should not be correctly decrypted even if there is merely a minor difference between the encryption key and decryption key.

If an encryption scheme satisfies the above-mentioned conditions then its key sensitivity is considered high. In our proposed scheme, we use BLAKE2 hash function to produce encryption key Key_1 and the two pseudo-random key-streams, since the output of BLAKE2 is very sensitive to changes occurred in the input parameter, even a minor change made in encryption key Key_1 or embedding key Key_2 will lead to significant variation in the produced pseudo-random key-streams, and alter all the following encryption operations. To verify the key sensitivity of our scheme for the first condition, we change the last bit of Key_1 to generate another encryption key, and Key_2 remains the same. The corresponding cipher-images of 'Baboon' encrypted by these two slightly different encryption keys are shown in Fig. 12, and their difference image is also presented. The chaotic difference image of these two cipher-images proves the fulfilment of key sensitivity analysis' first condition for our encryption method.

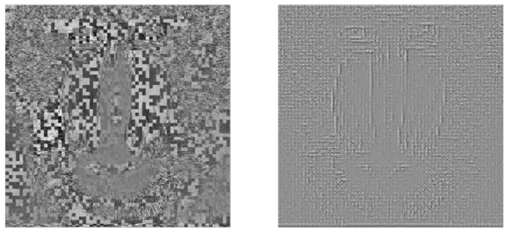
To evaluate the key sensitivity of the second condition,



(a) Proposed



(b) Ref [11]



(c) Ref [25]

Fig. 11: Direct replacement attack results (left is encrypted Baboon image, right is reconstructed Baboon image)

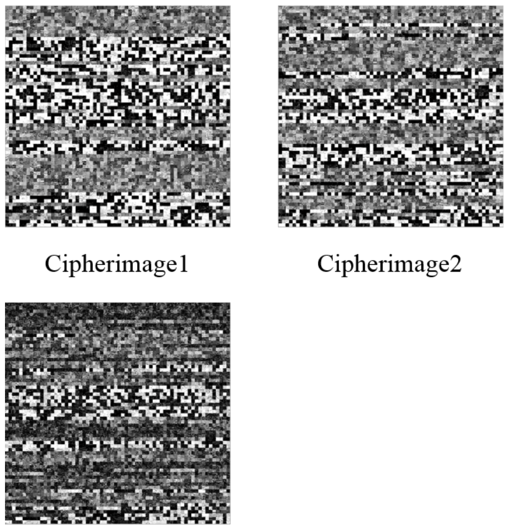


Fig. 12: Key sensitivity analysis for encryption process

we slightly modify Key_2 to generate another decryption key by changing its last bit, since Key_2 is the only key that is available to decoder. The decryption results of cipher Baboon image using these two different embedding keys are shown in Fig. 13. It is clear that even when there is a small variation in Key_2, correct decryption cannot be realized under our encryption/decryption scheme. Therefore, our proposed technique also satisfies the second condition of key sensitivity analysis.

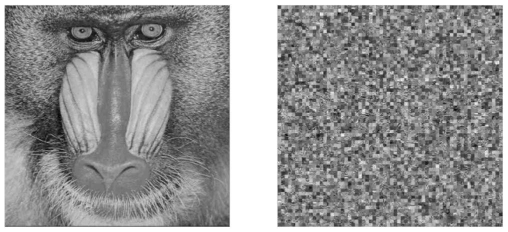


Fig. 13: Key sensitivity analysis for decryption process: left image decrypted by the original key, right image decrypted by the modified key

E. Differential attack

Differential attack is a chosen-plaintext attack, in which attacker makes a minor change in the chosen plain-image and observes the changes produced in cipher-image. By computing the difference between the chosen plain-images and the corresponding cipher-images, attacker tries to deduce a statistical relationship between them [32]. A cryptosystem is considered as resistance against the differential attack if one minor change in the plain-image causes large changes in the cipher-image. Two statistical evaluation parameters, net pixel change ratio (NPCR) and unified average change in intensity (UACI) [33], are widely used for checking the robustness of an image encryption scheme against differential attack. NPCR denotes the change rate of the number of pixels in the cipher-image while one pixel in plain-image is changed. The higher the value of NPCR, the more secure is the encryption scheme. UACI measures the average intensity difference between two cipher-images. To get higher security, the value of UACI should be close to 33%. The calculation of these two parameters involves a slight modification in one random pixel value of the plain-image. The plain-image and the slightly changed plain-image are then encrypted by the same encryption key. Supposing that the cipher-images before and after one random pixel changes in plain-image are denoted as C^1 and C^2 respectively, then NPCR and UACI can be calculated using the following formulas:

$$D(i,j) = \begin{cases} 0, if \ C^1(i,j) = C^2(i,j), \\ 1, if \ C^1(i,j) \neq C^2(i,j). \end{cases} \tag{7}$$

$$\text{NPCR} : N(C^1, C^2) = \frac{\sum_{i,j} D(i,j)}{H \times W} \times 100\%, \tag{8}$$

$$\text{UACI} : U(C^1, C^2) = \frac{\sum_{i,j} \frac{\left|C^1(i,j) - C^2(i,j)\right|}{255}}{H \times W} \times 100\%, \tag{9}$$

TABLE IX: Results of differential attack tests on various images

Image	Ref [11]		Ref [25]		Proposed	
	NPCR%	UACI%	NPCR%	UACI%	NPCR%	UACI%
Fruits	6.8665e-04	4.1887e-06	0	0	97.92	37.99
Girl	0	0	0	0	96.62	39.96
House	0	0	0	0	96.49	36.93
Baboon	0	0	0	0	97.80	39.48
Raffia	2.4033e-04	3.7399e-06	0	0	99.49	23.68
Goldhill	0	0	0	0	97.74	39.68
Lena	0	0	0	0	97.77	39.17
Peppers	1.7548e-04	8.5270e-07	0	0	98.06	39.27
Sailboat	1.6785e-04	7.0310e-07	0	0	97.27	39.99
Pentagon	0	0	0	0	99.43	21.97

NPCR and UACI analyses have been tested on various images by increasing its first pixel by 1. In Table IX, the NPCR and UACI values for images encrypted by our algorithm, *Algorithm-3* in [11], and [25] are listed, from which we can observe that our encryption scheme has a certain degree of defence capability against differential attack, while for *Algorithm-3* in [11] and [25]'s algorithm, the NPCR and UACI values are all almost zero, indicating their low diffusion property.

F. Statistical model-based attack

In statistical model-based attack, attacker studies the predictability of a particular element or the predictable relationship of some data segments between plain-image and cipher-image. The obtained relationship is used either to predict the plain-image without the knowledge of decryption key or to reduce the key search space to make the brute-force attack become easier [32]. Generally, histogram and correlation chart of the plain-image and cipher-image are two common ways to indicate the relationship. Histogram analysis illustrates the distribution of pixels in the image by counting the number of pixels at each gray scale level. The histogram of plain 'Baboon' and encrypted image under our scheme are shown in Fig. 14. It is clear that histogram of encrypted image has little statistical similarity to that of the plain image, but it 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 chart, we initially identify the neighbourhood of diagonal pixels from the plain-image and cipher-image. Then 1000 pairs of two diagonal adjacent pixels are randomly selected, and plot the correlation diagram based on the value of each pixel and its diagonal neighbour. The corresponding correlation charts of plain 'Baboon' and the corresponding cipher-image are presented in Fig. 15. We can observe that the linear property, which reflects the correlation degree between pixels, shown in cipher-image is a little less than that shown in plain-image, but it cannot be totally removed, also because we perform encryption in 8×8 blocks unit just like JPEG.

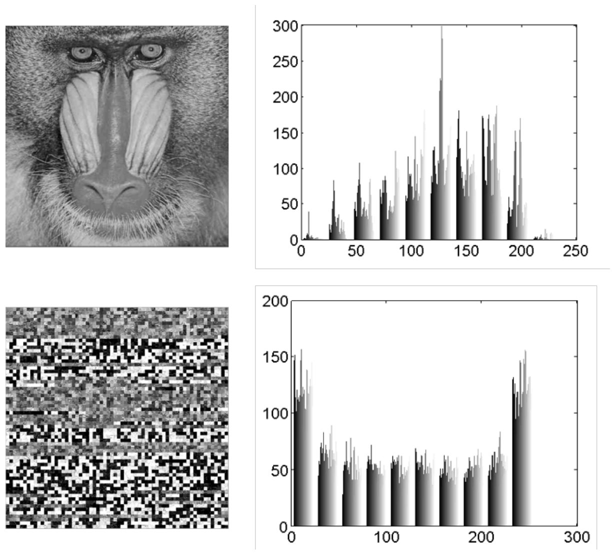


Fig. 14: Histogram charts of plain 'Baboon' image and encrypted image

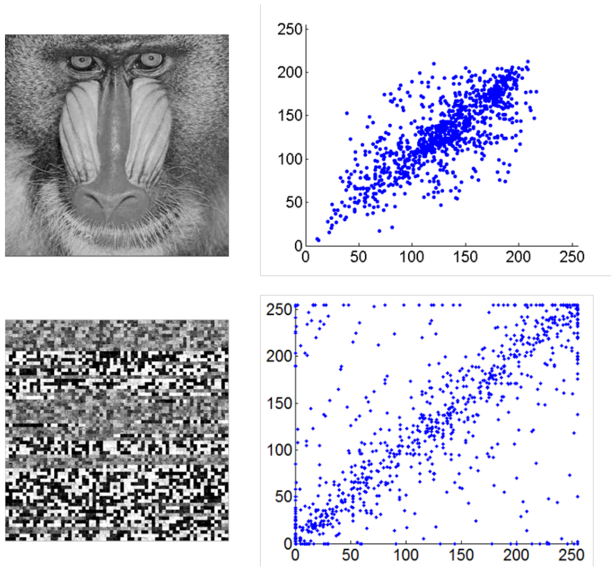


Fig. 15: Correlation charts of plain 'Baboon' image and encrypted image

V. Conclusion

In this paper, we propose a new joint image compression and encryption scheme, in which the encryption key is dependent on the plain-image. The whole scheme has good diffusion property, and is compression friendly to JPEG. To produce the image content-sensitive key, we adopt BLAKE2 hash function to produce the 256-bit encryption key. Our encryption operations include three parts: new orthogonal 8×8 transforms transformation, DC coefficients encryption, and AC encryption. We generate new transforms by introducing rotation angles into order-8 DCT's flow-graph structure, and apply them alternatively for 8×8 blocks' transformation, controlled by the encryption key. DC coefficients are encrypted after quantization by 8×8 blocks' permutation and XOR operation. For AC coefficients encryption, we embed the 256-bit encryption key into the bitstream of some certain AC coefficients, and the data embedding/extracting operation is controlled by another secret key, called embedding key. This data embedding strategy can reduce the cost of sending different 256-bit encryption keys to the decoder when different plain-images are compressed and encrypted. Extensive experiments and security analysis are conducted to show the good compression and encryption performance of our proposed encryption scheme.

There are few extensions we believe that can be considered in our following work. First, currently the encryption scheme cannot achieve perfect correlation removal effect and diffusion property, because the whole encryption work is realized in 8×8 block unit. Hence, in the next step, we may try to do transformation in different block sizes and to execute permutation in some smaller units, perhaps 4×4 blocks permutation or image pixels' bit-plane level permutation. Additionally, our encryption scheme is mainly realized in the transformation stage and entropy coding stage of JPEG, other existing JPEG encryption methods, such as zig-zag scanning order change or quantization table change, can be combined with our current proposed encryption scheme, in order to enlarge the encryption space by one dimension. Finally, we will try to extend our current encryption operations for other image/video compression standards, such as JPSEC standard and MPEG-4.

References

[1] N. K. Pareek, "Design and analysis of a novel digital image encryption scheme," *arXiv preprint arXiv:1204.1603*, 2012.

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

[3] S. P. Indrakanti and P. Avadhani, "Permutation based image encryption technique," *International Journal of Computer Applications (0975–8887) Volume*, 2011.

[4] M. A. B. Younes and A. Jantan, "An image encryption approach using a combination of permutation technique followed by encryption," *International journal of computer science and network security*, vol. 8, no. 4, pp. 191–197, 2008.

[5] A. Mitra, Y. S. Rao, S. Prasanna *et al.*, "A new image encryption approach using combinational permutation techniques," *International Journal of Computer Science*, vol. 1, no. 2, pp. 127–131, 2006.

[6] H. Somani, N. Tiwari, M. Chawla, and M. Shandilya, "Image encryption using block shuffling and affine transform: A review," *International Journal of Computer Applications*, vol. 95, no. 19, 2014.

[7] A. Nag, J. P. Singh, S. Khan, S. Ghosh, S. Biswas, D. Sarkar, and 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.

[8] A. Jolfaei, X.-W. Wu, and V. Muthukkumarasamy, "On the security of permutation-only image encryption schemes," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 235–246, 2016.

[9] C. Li and K.-T. Lo, "Optimal quantitative cryptanalysis of permutation-only multimedia ciphers against plaintext attacks," *Signal processing*, vol. 91, no. 4, pp. 949–954, 2011.

[10] S. Li, C. Li, G. Chen, N. G. Bourbakis, and K.-T. Lo, "A general quantitative cryptanalysis of permutation-only multimedia ciphers against plaintext attacks," *Signal Processing: Image Communication*, vol. 23, no. 3, pp. 212–223, 2008.

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

[12] Z. Qian, X. Zhang, and S. Wang, "Reversible data hiding in encrypted jpeg bitstream," *IEEE transactions on multimedia*, vol. 16, no. 5, pp. 1486–1491, 2014.

[13] 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.

[14] Z. Qian, H. Zhou, X. Zhang, and W. Zhang, "Separable reversible data hiding in encrypted jpeg bitstreams," *IEEE Transactions on Dependable and Secure Computing*, 2016.

[15] C.-P. Wu and C.-C. Kuo, "Design of integrated multimedia compression and encryption systems," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 828–839, 2005.

[16] A. Massoudi, F. Lefebvre, C. De Vleeschouwer, B. Macq, and J.-J. Quisquater, "Overview on selective encryption of image and video: challenges and perspectives," *EURASIP Journal on Information Security*, vol. 2008, no. 1, p. 1, 2008.

[17] J. Zhou, Z. Liang, Y. Chen, and O. C. Au, "Security analysis of multimedia encryption schemes based on multiple huffman table," *IEEE Signal Processing Letters*, vol. 14, no. 3, pp. 201–204, 2007.

[18] G. Jakimoski and K. Subbalakshmi, "Cryptanalysis of some multimedia encryption schemes," *IEEE Transactions on Multimedia*, vol. 10, no. 3, pp. 330–338, 2008.

[19] Y. Zhang, D. Xiao, H. Liu, and H. Nan, "Gls coding based security solution to jpeg with the structure of aggregated compression and encryption," *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, no. 5, pp. 1366–1374, 2014.

[20] R. Bose and S. Pathak, "A novel compression and encryption scheme using variable model arithmetic coding and coupled chaotic system," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 4, pp. 848–857, 2006.

[21] H. Kim, J. Wen, and J. D. Villasenor, "Secure arithmetic coding," *IEEE Transactions on Signal processing*, vol. 55, no. 5, pp. 2263–2272, 2007.

[22] M. Saarinen and J. Aumasson, "The blake2 cryptographic hash and message authentication code (mac)," Tech. Rep., 2015.

[23] S. Bahrami and 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*, vol. 124, no. 18, pp. 3693–3700, 2013.

[24] W.-H. Chen, C. Harrison, and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *communications, IEEE Transactions on*, vol. 25, pp. 1004–1011, 1977.

[25] P. Li and K.-T. Lo, "Joint image compression and encryption based on alternating transforms with quality control," in *2015 Visual Communications and Image Processing (VCIP)*. IEEE, 2015, pp. 1–4.

[26] L. Wang and M. Goldberg, "Progressive image transmission by transform coefficient residual error quantization," *IEEE transactions on communications*, vol. 36, no. 1, pp. 75–87, 1988.

[27] W. Cham and R. Clarke, "Application of the principle of dyadic symmetry to the generation of orthogonal transforms," *Communications, Radar and Signal Processing, IEE Proceedings F*, vol. 133, no. 3, pp. 264–270, 1986.

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

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

[30] K. Minemura, K. Wong, R. 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*, 2016.

[31] B. Schneider, *Applied Cryptography: Protocols, algorithms, and source code in C*. John Wiley & Sons, 1996.

[32] N. Taneja, B. Raman, and I. Gupta, “Combinational domain encryption for still visual data,” *Multimedia tools and applications*, vol. 59, no. 3, pp. 775–793, 2012.

[33] Y. Wu, J. P. Noonan, and S. Agaian, “Npcr and uaci randomness tests for image encryption,” *Cyber journals: multidisciplinary journals in science and technology, Journal of Selected Areas in Telecommunications (JSAT)*, pp. 31–38, 2011.