JOINT IMAGE COMPRESSION AND ENCRYPTION SCHEMES

PEIYA LI

PhD

The Hong Kong Polytechnic University

2018

The Hong Kong Polytechnic University

Department of Electronic & Information Engineering

Joint Image Compression and Encryption Schemes

Peiya Li

A thesis submitted in partial fulfilment of the requirements

for the degree of Doctor of Philosophy

January, 2018

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Peiya Li (Name of Students)

Abstract

Thanks to the rapid growth of computer networks and information technology, the number of multimedia applications related to images is rapidly increasing, thus the security issue of image storage and transmission becomes much more essential. Many encryption algorithms have been proposed to ensure images' safety. Among these encryption schemes, strategies of integrating image compression process with encryption algorithms have received much attention, since compression is a must-do step for most images we see on the Internet. Among various compression standards, JPEG is the most commonly applied method for lossy compression of digital images. This thesis presents our contributions in realizing image encryption during the JPEG compression process.

In this thesis, four different joint image compression and encryption schemes are proposed by introducing encryption techniques into the intermediate stages of JPEG. All the encryption algorithms are format compliant to JPEG. The first encryption scheme is realized by applying new order-8 orthogonal transforms for blocks' transformation, followed by block permutation on quantized 8×8 blocks. These new transforms are produced by embedding rotation angle of π to 8×8 DCT's flow-graph structure. Through controlling the number of rotation angles embedded, we can achieve the quality control of the final encrypted images. In the second encryption scheme, we improve the first scheme by allowing more rotation angles, not just π, to be embedded into the flow-

graph of DCT, and the coding efficiency of these newly generated transforms is better than that of the transforms used in the first scheme. Additionally, we enhance the whole cryptosystem's diffusion property by applying plain-image-dependent key for encryption, and propose a new data hiding technique for concealing the encryption key. The encryption scheme is JPEG compression friendly, can achieve a good balance between the encryption power and the compression efficiency. In the third encryption scheme, we realize joint image compression and encryption by using 16×16 DCT, in which the encryption techniques are mainly conducted at JPEG's transformation stage and entropy coding stage. To enhance cryptosystem's diffusion property, we also establish a relationship between the secret key and the plain-image, like the second scheme. For the transformation stage encryption, we segment the plain-image into non-overlapping 16×16 blocks, and use order-16 DCT for transformation. Then we distribute all the coefficients of 16×16 transformed blocks into 8×8 blocks, and do block permutation and DC coefficients confusion. For the entropy coding stage encryption, we shuffle all Run/Size and Value (RSV) pairs of AC coefficients, and embed end-of-block (EOB) identifiers to keep format compliance. As for the fourth encryption scheme, we use adaptive block-size (8×8 size block and 16×16 size block) for JPEG image encryption. Given a plain-image, we first segment it into 8×8 size blocks or 16×16 size blocks, according to two different segmentation schemes. Then we transform these different size blocks using the corresponding size of DCT, followed by the 8×8 blocks permutation and same run value RSV pairs' shuffling operation. The fourth scheme has comparable compression efficiency as the second scheme, and can achieve better statistical attack defense ability than the second scheme.

List of Publications

Journal papers:

1. Li, Peiya, and Kwok-Tung Lo, "A Content-Adaptive Joint Image Compression and Encryption Scheme." *IEEE Transactions on Multimedia*, 2017 (IF=3.509).

2. Li, Peiya, and Kwok-Tung Lo, "Joint image compression and encryption based on order-8 alternating transforms." *Journal of Visual Communication and Image Representation*, vol. 44, pp. 61-71, 2017 (IF=2.164).

Conference paper:

1. Li, Peiya, and Kwok-Tung Lo, Joint image compression and encryption based on alternating transforms with quality control, in *Visual Communications and Image Processing (VCIP)*, IEEE, 2015, pp. 14.

Acknowledgements

The past three years of studying at the Hong Kong Polytechnic University is the most precious moment of my life. There are many people who have offered help and guidance to me and have inspired me. I would like to express my sincere gratitude to them.

First and foremost I want to thank my supervisor Dr. Kwok-Tung Lo, who provides me with guidance and countless assistance throughout my PhD study. I appreciate all his contributions of time and ideas to make my PhD experience productive and stimulating. Besides, I have a great admiration for the enthusiasm and rigorous attitude he has for research, which will motivate me in the future life. In a word, this thesis could not have been finished without him.

I would also like to express thanks to my lab-mates for their invaluable assistances and friendships. Kwok-Tung Sze welcomed me to the lab, who is also a PhD student of Dr. Kwok-Tung Lo. He is very humorous and optimistic, and has shared with me many research resources and personal experiences. Zhenhui Situ, Shuangnan Liu and Junkai Chen have given me a lot of advice in preparing my research reports and examinations, they are really very patient, and warm-hearted. Additionally, I thank the staff at Department of Electronic and Information Engineering for their patience help and the postgraduate scholarship from The Hong Kong Polytechnic University for its financial help.

Last but certainly not the least, I would like to thank my parents for all their love and encouragement. It was under their watchful eye that I obtained so much drive and an ability to address challenges head on. Their unconditional love is most important aspect in my whole life.

Table of contents

List of figures

List of tables

List of abbreviations

AES Advanced Encryption Standard

BPP bit per pixel

CBC Cipher-Block Chaining

CFB Cipher Feedback

CTR Counter

DCT discrete cosine transform

DES Data Encryption Standard

DPCM differential pulse code modulation

EAC energy of AC coefficients

ECB Electronic Codebook

EOB end-of-block

ETC Encryption-then-Compression

IDCT inverse discrete cosine transform

KSA key-scheduling algorithm

LFSR Linear Feedback Shift Registers

NCC nonzero coefficients count

NZCA non-zero-counting attack

OFB Output Feedback

PLZ position of last nonzero coefficients

PRGA pseudo-random generation algorithm

PSNR peak signal-to-noise ratio

QF quality factor

RC4 Rivest Cipher 4

RSV Run/Size and Value

SSIM structural similarity

VLC variable-length codes

VoD Video on Demand

Chapter 1

Introduction

In recent years, due to the drastic development of network technology, approaches for communication have been taken to a new era. People communicate with each other anywhere, any-time through various devices, such as laptops, personal computers, and smart mobile phones. Using these devices, multimedia content can be easily generated and distributed to the specific people/group through Internet [1]. However, this easy access mode and distribution convenience also increase the risk of eavesdropping and intercepting [2]. In major parts of these multimedia data (image, video, or audio), no matter whether they are private or confidential, they need security mechanisms to offer various degree of protection [1, 3]. The common solution to protect digital data from eavesdropping and intercepting is to either encrypt the message or encrypt the channel, and both of them need some knowledge of cryptography [4]. 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 digital image data is receiving more and more attention.

This chapter first presents some fundamentals of image encryption, which include features of image data that make image encryption different from text data encryption, and criteria for evaluating and comparing image cryptosys-

tems. Then, motivations and contributions of this thesis are explained. Finally, the outline of the thesis is presented.

1.1 Fundamentals of Image Encryption

Image encryption tries to convert original image to another image that is hard to understand, and ensure that nobody could get to know the plain-image's content without a key for decryption. Since the 1970s, many cryptographic techniques have been developed, and some of them, such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES) [5], have been widely adopted as the standardized encryption algorithms [6]. However, unlike text message, image data has some unique features, which make some conventional encryption algorithms working well for text data encryption become not so suitable for image encryption [7]. Hence, in order to design a good image cryptosystem, we should first analyse the differences between text data and image data, and some major differences are listed as follows.

- For digital images, the size of them are generally very large, even if they are efficiently compressed. Since the encryption speed of some traditional ciphers is not sufficiently fast, especially for software implementation, if they are adopted for encrypting images, the processing time will be much longer.

- For text data, its cipher-text must be decrypted to the original plain-text in a full lossless manner. However, for image data, its cipher-image can be decrypted to the original plain-image in some lossy manner.

- For uncompressed image, there exists high information redundancy, which may make block ciphers running in Electronic Codebook (ECB) mode [8] fail to conceal all visual information of plain-images.

- For digital images, they are usually represented as two-dimensional (2D) arrays, while for text data, they are represented as a sequence of words. If text encryption algorithms are applied, these 2D arrays of data must be converted to 1D arrays of data first, which will increase the processing time.

After an image encryption scheme is proposed, how to judge its performance is also very important. Generally, there are a number of criteria that can be used to evaluate the efficiency and security of an image encryption scheme.

- *Visual degradation.* This criterion evaluates the cipher-image's perceptual distortion degree with respect to the plain-image [9,10]. It supposes that users can still decode and view the cipher-image even without the decryption key. Different applications may have different visual quality requirements of cipher-images. For pay-after-trial services of multimedia data, like pay-TV, Video on Demand (VoD) [11], they may prefer to encrypt multimedia data in a way to make encrypted content partially perceptible even without the decryption key, and this down-graded content can serve as a preview chance to attract potential customers. For military or financial applications, multimedia data may need to be encrypted completely, which means high visual degradation, so that zero visual information can be recognized by an unauthorized person. Therefore, when we use this criterion for evaluating image encryption schemes, we should consider their application scenarios. The peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [9, 12] are two common metrics used to measure visual degradation.

- *Cryptographic security.* This represents the encryption scheme's resistance ability against various cryptanalysis techniques, such as brute-force

attack, key sensitivity analysis, replacement attack, differential attack, and statistical attack. We will explain these attacking methods in following chapters.

- *Format compliance (also called syntax-awareness [13], transcodability [11] or transparency).* Format information is generated after image data is encoded using some compression algorithms. This information will be used by decoder to successfully recover the compressed data and to keep the communication synchronized between the encoder and decoder [14]. When applying encryption algorithms to images, the encrypted bit-stream should be compliant with the compressor, which means that the encrypted bit-stream should be decoded by any standard decoder even without decryption. This property is essential since it can preserve some good features of the corresponding compression standard [15].

- *Compression friendliness.* In today's resource constrained environment, efficient compression techniques are required to process bulky multimedia data, in order to save the transmission bandwidth and storage space. Thus in all cases, when we use encryption algorithms to ensure security of the compressed data, these algorithms should have no or very little influence on the data compression efficiency. Some algorithms may introduce some overheads which are necessary for decryption, but the impact of these overheads on the final compression ratio should be limited [15, 16].

- *Encryption efficiency.* Encryption operation should be highly efficient because the size of multimedia data is often very huge and the number of users is extremely large. And this property is particularly important for image applications with real-time processing requirement [16].

1.2 Research Motivations and Contributions

Nowadays, since most of the images we see on Internet are compressed, study on how to offer protection for these compressed images has become crucial. Given a standard image compressor, researchers generally conduct encryption operations at three different positions, before-compression, in-compression, and after-compression. In order to save the computation cost for image encryption and compression, this thesis mainly focuses on joint image compression and encryption scheme's design, which is to realize image encryption during the compression process.

Among various image compression standards, JPEG is the most commonly used method that can greatly save the storage space and transmission bandwidth. Hence, study on joint JPEG image compression and encryption has important theoretical and practical values, and all image encryption schemes proposed in this thesis are based on lossy JPEG standard. In Section 1.1, criteria for evaluating image encryption schemes have been explained, however, some of them cannot be achieved simultaneously, such as cryptographic security and compression friendliness. For joint image compression and encryption schemes, high cryptographic security is usually achieved in the cost of sacrificing compression efficiency, while compression friendly to the corresponding image compressor generally means the cryptosystem cannot obtain fully confidentiality.

In this thesis, all proposed joint image compression and encryption schemes aim to at least meet the "visual degradation" and "format compliance" criteria. For other three criteria (cryptographic security, compression friendliness, and encryption efficiency) mentioned in Section 1.1, different proposed schemes have different focuses. Totally, four joint JPEG image compression and encryption schemes are proposed in this thesis, and they are described as

follows.

- A joint JPEG image compression and encryption scheme based on order-8 alternating transforms is proposed. This work does not aim to achieve fully confidentiality, since it only performs encryption operation at JPEG's transformation stage. The major contribution of this work is to develop new order-8 orthogonal transforms with coding efficiency similar to discrete cosine transform (DCT), then apply them alternatively for 8×8 blocks' transformation according to a predefined secret key. These transforms are generated by embedding extra rotation angles to 8×8 DCT's flow-graph, and by carefully controlling the number of rotation angles embedded, the quality control of encrypted images can also be achieved. Experimental results have confirmed that the proposed scheme is compression friendly to JPEG, has good encryption efficiency, and can achieve a certain level of security. More details of this scheme are presented in Chapter 3.

- A content-adaptive joint image compression and encryption scheme is proposed, which aims at encryption power's enhancement, on the premise of maintaining JPEG's compression efficiency. The scheme is plain-image-content-adaptive, since the secret encryption key is generated from the plain-image using BLAKE2 hash algorithm. Three encryption operations are included in the cryptosystem, alternating new orthogonal transforms transformation, DC coefficients encryption, and AC coefficients encryption. To save the cost of transmitting different encryption keys each time to decoder for decryption when plain-image changes, the encryption key is embedded into the entropy encoded bitstream of some AC coefficients, and the whole embedding procedure is controlled by another secret key, called embedding key. Experimental results have shown

that the proposed cryptosystem is efficient and has good compression and security features. More details of this cryptosystem are described in Chapter 4.

- Two joint image encryption and compression schemes based on 16×16 DCT are proposed, where one scheme emphasizes compression performance, another highlights cryptographic security. The major objectives we want to achieve in these two schemes are to ensure the format of the final encrypted and compressed bitstream is compliant to JPEG, and to improve image cryptosystem's confusion and diffusion properties, on the premise of not compromising JPEG's compression efficiency too much. To enhance cryptosystem's diffusion property, in both schemes, a relationship between the secret key and the plain-image is established through the hash function BLAKE-256. In the first encryption scheme, encryption operations are conducted at JPEG's transformation stage and quantization stage, which include 16×16 blocks' transformation using order-16 DCT, 8×8 blocks' permutation, and DC coefficients confusion. As for the second scheme, after finishing all encryption operations contained in the first scheme, another encryption operation is added at JPEG's entropy coding stage by shuffling all Run/Size and Value (RSV) pairs of AC coefficients. Performance evaluation on these two encryption schemes have shown that the first scheme has better compression efficiency, while the second scheme has better defense ability against the differential attack and statistical attack. More details of these two schemes are introduced in Chapter 5.

- A JPEG image encryption scheme using adaptive block-size is proposed, in which encryption is realized at JPEG's transformation stage and entropy coding stage. The major objective of the proposed scheme is not

to offer full confidentiality, but tries to provide the compressor with certain level of protection ability, and at the same time not compromising JPEG's compression performance too much. Based on two different segmentation criteria, the plain-image is initially segmented into two different sizes of blocks, 8×8 and 16×16, then the corresponding size of DCT is applied for these blocks' transformation. After that, 8×8 blocks' permutation and RSV pairs with same run value shuffling are conducted to encrypt the plain-image. This scheme's protection power is influenced by the quality factor which controls the compression ratio, higher quality factor value results in stronger protection power. More details of this scheme are present in Chapter 6.

1.3 Thesis Outline

The thesis consists of seven chapters, and they are outlined as follows.

Chapter 2 introduces some basic knowledge of modern cryptology, which includes cryptography and cryptanalysis. A brief description on how the JPEG algorithm works and existing image encryption schemes related to JPEG standard are also presented in this chapter.

Chapter 3 presents how the first joint JPEG image compression and encryption scheme is realized. The way for producing new order-8 orthogonal transforms is explained using mathematical formulas, and the encryption algorithm using these transforms is also introduced in this chapter. Additionally, to improve the robustness against the direct replacement attack, Fisher-Yates shuffle is adopted in this scheme to randomly change the positions of 8×8 blocks, because of its simple implementation. Extensive experiments are conducted to show the good security and compression performance of the proposed encryption scheme.

Chapter 4 presents details for realizing the content-adaptive joint image compression and encryption scheme. Two secret keys, named encryption key and embedding key, are used in this scheme, where the encrypted key is produced from the plain-image by taking it as the input of BLAKE2 hash function, and the embedding key is a predefined secret seed, only known to the encoder and decoder. Implementations of the three encryption operations (new orthogonal transforms, DC coefficients encryption, and AC coefficients encryption) are introduced in this chapter. Extensive experimental results are reported to show the proposed method is compression friendly and format compliant to JPEG, along with a detailed security analysis to illustrate the scheme's persistence to various cryptanalysis strategies.

Chapter 5 presents how our two order-16 DCT-based encryption schemes are developed. The hash function BLAKE-256 and Chen's chaotic system are adopted for producing the pseudo-random key-streams, which will be used to control all encryption operations. The reason why BLAKE-256 is used is because of its high security and fast speed. While for Chen's chaotic system, it has abundant and complex dynamic behaviors. In the first scheme, encryption is realized during JPEG's transformation stage and after the quantization stage by replacing original order-8 DCT with order-16 DCT for block's transformation, followed with coefficients distribution, block permutation, and DC coefficients confusion. The added encryption operation, RSV pairs shuffling, in the second scheme is also introduced. Detailed performance evaluations of these two encryption schemes are presented in this chapter to show the first scheme's good compression performance and the second scheme's encryption power.

Chapter 6 presents how the adaptive-block size encryption scheme is realized. Two image segmentation methods, key-based segmentation and image-based segmentation, are first explained, which will be used to split the plain-

image into 8×8 blocks and 16×16 blocks. Based on which segmentation criterion is adopted, two different encryption schemes are produced. Both schemes contain four encryption operations: plain-image segmentation, coefficients distribution, block permutation, and same run RSV pairs' shuffling. The difference between these two schemes is that for the scheme using image's information to control segmentation, it produces extra segmentation data, hence it needs two keys to realized the whole encryption procedure. One key is used to control the above four encryption operations, another key is used to encrypt the extra segmentation data. Both schemes retain JPEG's good compression performance, and offer a sufficient level of image protection.

Chapter 7 concludes the research work presented in this thesis and discusses some potential directions for future research.

Chapter 2

Literature Review

In this chapter, basic knowledge of cryptology, consisting of cryptography and cryptanalysis, is firstly presented, then the work flow of JPEG is introduced. The different positions for embedding encryption operations into JPEG's compression process result in three types of JPEG image encryption algorithms, pre-compression encryption algorithm, in-compression encryption algorithm, and post-compression encryption algorithm. Some related works of these three categories are summarized in this chapter.

2.1 Cryptology

Cryptology contains cryptography and cryptanalysis, where cryptography is the study of techniques applied to encrypt data, and cryptanalysis is the study of techniques used to break existing cryptosystems. Cryptography and cryptanalysis are greatly dependent on each other.

2.1.1 Cryptography

The many schemes used for encryption constitute the area of study known as cryptography [17]. An encryption scheme is also called a cipher, or a cryp-

tosystem. In Figure 2.1, the general structure of a cipher is given. Message sent for encryption is called plain-text, and the encrypted message is called cipher-text [18]. We denote the plain-text and the cipher-text by P and C, respectively. The encryption procedure of a cipher can be described as $E(P, Key_e) = C$, where $E(\cdot)$ is the encryption algorithm, and Key_e is the encryption key. Similarly, the decryption procedure can be described as $D(C, Key_d) = P$, where $D(\cdot)$ is the decryption algorithm, and Key_d is the decryption key. In the 19^{th} century, Kerchhoff proposed that the encryption algorithms were supposed to be known to the attackers, the security of a cipher, therefore, should rely on the secrecy of the encryption/decryption key instead of the encryption algorithm itself [19]. Different types of ciphers can be obtained when different classification criteria are adopted.

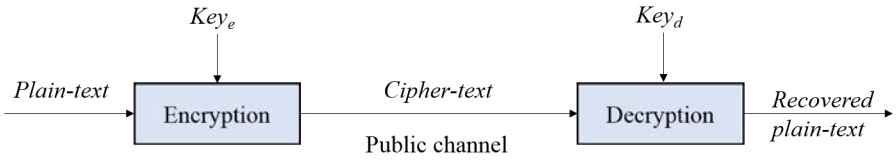


Figure 2.1: Encryption and decryption of a cipher.

According to the relationship of Key_e and Key_d, there are two kinds of ciphers. If the encryption key Key_e is equal to the decryption key Key_d, the cipher is called a private-key cipher or a symmetric cipher [20]. Under this situation, the receiver must transmit the encryption/decryption key to the receiver through a secret and separate channel. When the two keys are not equal, the cipher is called a public-key cipher or an asymmetric cipher [20]. For asymmetric ciphers, the encryption key is published, while the decryption key must be kept private, and we do not need any additional secret and separate channel for key transfer. Both ciphers have their advantages and disadvantages. For symmetric ciphers, they are generally fast, but every participated party must ensure the key absolutely secret, and this leads to enlarged secu-

rity vulnerability with the number of involved parties increasing. Moreover, for each pair of persons who want to communicate using a different key must be agreed upon, and this makes the key management work very cumbersome. For asymmetric ciphers, pairs of keys are used (public key and private key). Their computational costs are much more expensive, but the public key can be distributed to every one, only private key that is known only to the owner needs to be protected.

According to the encryption structure, encryption algorithms can be classified into block ciphers and stream ciphers. Block ciphers encrypt the plain-text block by block, and each block is mapped into another block with the same size. Block ciphers can be run in different operation modes, and each of these modes provides different encryption/decryption effect, allowing users to choose appropriate modes to meet their own applications' requirements. The most common modes are: Electronic Codebook (ECB), Cipher-Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) [8]. Block ciphers are mixed arithmetic operation, which can provide non-linearity. And larger block size means more secure of the cipher, but at the same time, the encryption and decryption algorithms become more complex. Unlike block ciphers, stream ciphers do not transform blocks of data to another block of data. Instead, they encrypt the plain-text with a pseudo-random sequence (called key-stream) produced from the encryption key. In a stream cipher, each plain-text digit is encrypted one at a time with the corresponding digit of the key-stream, usually by the XOR operation, to give a digit of the cipher-text stream. Linear Feedback Shift Registers (LFSR) and Rivest Cipher 4 (RC4) are two typical examples of stream ciphers. To ensure the security of a stream cipher, the key-stream used must have a large period and deriving the cipher's key or internal state from the key-stream must be impossible.

According to the percentage of data encrypted, encryption can be divided into two types: full encryption and partial encryption (also called selective encryption). Partial encryption is often accompanied with compression algorithms, since compression algorithms can decompose data into a number of different logical parts with different significance [21].

2.1.2 Cryptanalysis

Cryptanalysis is the science of analysing and breaking an encrypted message as a whole or in part when the decryption key is not available. An attempted cryptanalysis is called an attack. Depending on the amount of known information and the amount of control over the system by the attack, four typical attacks can be obtained: cipher-text-only attack, known-plain-text attack, chosen-plain-text attack, and chosen-cipher-text attack [2, 18].

- *Cipher-text-only attack.* This attack is the most realistic and basic attacking method in which attackers can only obtain the encrypted data. By observing cipher-texts of several plain-texts encrypted with the same key, the cryptanalyst tries to recover the corresponding plain-text or the encryption key. All cryptosystems should be designed to withstand this type of attack. One of the typical methods for cipher-text-only attack is to try all possible keys in the brute-force manner, which is the brute-force attack. To make brute-force attack feasible, the key space of a cryptosystem should be large enough.

- *Known-plain-text attack.* In this type of attack, attackers have access to the cipher-text and associated plain-text for several messages. They try to use these data to deduce the key, or to develop an algorithm to decrypt any new messages encrypted with the same key.

- *Chosen-plain-text attack.* In this case, attackers are allowed to choose the plain-text, and put it into the black box which contains the encryption algorithm and the encryption key. The black box outputs the corresponding cipher-text, and attackers can use the accumulated information about the plain-text and cipher-text pairs to deduce the secret key or at least a part of it. The attackers' goal is same as that in a known-plain-text attack.

- *Chosen-cipher-text attack.* Here, attackers can choose cipher-text, and put it into the black box that contains the decryption algorithm and the decryption key. The black box outputs the corresponding plain-text, and attackers try to obtain the secret key or a part of the key through studying the accumulated cipher-text and plain-text pairs.

- *Key sensitivity analysis.* An ideal cryptosystem should be extremely sensitive to the key used in the encryption/decryption algorithm, and it can be observed in two ways: (i) completely different cipher-images should be produced when slightly different encryption keys are used to encrypt the same plain-image; (ii) the cipher-image should not be correctly decrypted even if there is a minor difference in the encryption and decryption keys.

- *Statistical attack.* In this type of attack, attackers try to predict the plain-image without the knowledge of key through studying the predictable relationship of some data segments between plain-image and cipher-image. Generally, histograms and correlation charts are two common ways to measure the relationship. It is obvious that most natural images have high correlation between adjacent elements, hence if an encryption scheme can reduce such correlation, the relationship be-

tween plain-image and cipher-image will be decreased, and the encryption scheme is deemed to be efficient.

2.2 JPEG compression standard

JPEG [22] and JPEG2000 [23] are two image compression standards created by the Joint Photographic Experts Group committee, in which JPEG2000 was proposed in 2000, using wavelet transform for compression, later than JPEG, with the intention of superseding their original DCT-based JPEG standard. Nonetheless, as of 2017, the majority of image format used by various social network platforms as well as digital cameras is still lossy JPEG, and the JPEG Committee has recently initiated a new activity called JPEG Privacy & Security [24] to develop various security functions in lossy JPEG in order to realize secure image information sharing. Therefore, all security mechanisms proposed in this thesis are based on the lossy JPEG standard. In Figure 2.2, the general architecture of JPEG encoder is given, from which we can observe that it mainly contains four stages: DCT transformation stage, quantization stage, zigzag scan stage, and entropy coding stage [25]. For JPEG decoding process, stages contained are just the reverse of the four stages in encoding procedure.

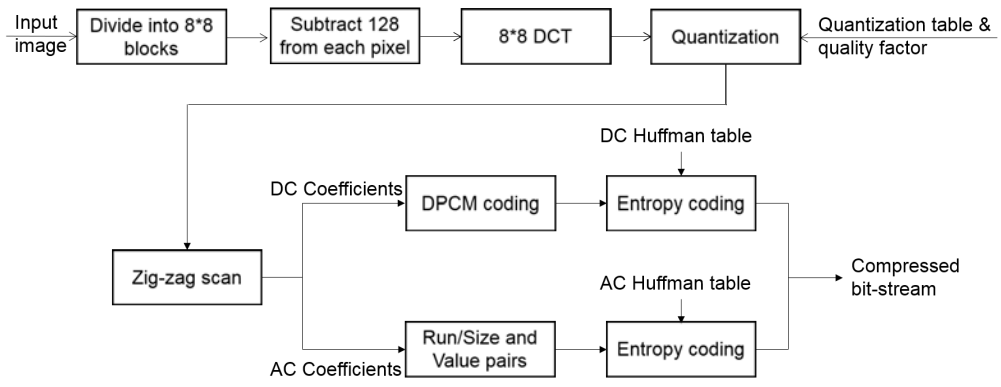


Figure 2.2: General architecture of JPEG encoder.

The whole compression process of JPEG can be described as: for a given plain-image, it is initially divided into non-overlapping 8×8 sub-blocks, and the gray values of all pixels subtract 128. Transform each 8×8 block into frequency domain by order-8 DCT, and range of DCT coefficients is $[-1024, 1023]$ [26]. Then these coefficients are quantized with respect to the quantization table scaled by quality factor (QF). After quantization, these quantized coefficients are zigzag scanned and the DC & AC coefficients are coded separately. For DC coefficients, differential pulse code modulation (DPCM) is used where only DC differential values are encoded. For AC coefficients, they are encoded in the form of RSV pairs, denoted as (r, v), where r represents the number of consecutive zeros existed before a non-zero AC coefficient, and v defines the non-zero AC coefficient. All (r, v) pairs and DC differences are further entropy coded into binary sequences using specific Huffman tables.

2.3 Existing JPEG image encryption algorithms

Considering the different potential positions for the encryption algorithms to be embedded into the compression process, JPEG image encryption can be divided into three categories: pre-compression encryption, in-compression encryption, and post-compression encryption.

2.3.1 Pre-compression encryption algorithms

Pre-compression encryption means performing encryption before compression, and decompression before decryption. Encryption algorithms, like the permutation-only encryption methods [27–36] or some chaos-based encryption schemes [37–43] which are directly operated on raw images, can be classified into this category. The permutation-only encryption schemes can be easily implemented and are available in both spatial and frequency domain, but they still have

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 [44–48]. In [49], Pareek *et al.* proposed an image cipher for encryption of gray images based on substitution, permutation, and a number of other well-known procedure. A total of sixteen rounds were used in their algorithm and in each round, the plain-image block size was kept secret key dependent. This proposed image cipher had sufficient large key space, was capable of resisting the brute-force attack, however, its secret key was deduced by a chosen-ciphertext attack [50]. In [51, 52], block scrambling-based image encryption schemes had been proposed for Encryption-then-Compression (ETC) systems, in which an image with $X \times Y$ pixels was first divided into non-overlapped blocks with $B_x \times B_y$, then four block-based processing steps (block scrambling, block rotation and inversion, negative-positive transformation, and color component shuffling) were applied to the divided image. Key space of these schemes are generally large enough against the brute-force attack, however, the encrypted image has almost the same correlation among pixels in each block as that of the original image, attackers can utilize the correlation to decrypt the image in some way. In [53], they regarded the encrypted images by [51, 52] as jigsaw puzzles, then decrypting cipher-images was similar to assembling the jigsaw puzzle. The jigsaw puzzles solve proposed in [53] could assemble the cipher-images in the schemes of [51, 52] to be approximately 80%, when the block size $B_x \times B_y$ was 28×28. In [54–57], Zhang *et al.* proposed methods to compress encrypted images with auxiliary information, but their encryption methods could not be integrated into JPEG's compression process, and the

compression efficiency that their schemes could achieve was lower than JPEG achieves. Generally, pre-compression encryption algorithms are inherently format compliant, but they are not quite suitable for lossy compression, since under lossy compression, pixel value in cipher-image cannot be fully recovered to the same value as the plain-image. Additionally, performing encryption before compression often destroys the correlation in plain-image, which is the major factor that determining whether digial image compression can be realized or not. Therefore pre-compression encryption schemes are usually not compression friendly.

2.3.2 In-compression encryption algorithms

In-compression encryption is to perform encryption and compression jointly, which is mainly realized through introducing encryption techniques into one stage or several stages of the underlying compressor. Encryptions implemented at various intermediate stages are listed as follows:

- *Encryption at transformation stage.* Encryption realized in transformation stage is mainly through replacing DCT with other orthogonal transforms [58–62] or encrypting DCT transformed coefficients [63–67]. In 1998, Shi and Bhargava [63] realized image encryption by encrypting the sign bits of DCT coefficients in JPEG, but even without the encryption key, however, useful image information can still be recovered by assigning all DC coefficients to 128 and all AC coefficients to positive [68]. Moreover, in [69], they also claimed that this AC coefficient sign encryption method was not secure enough. Since after DCT transformation, the 8×8 DCT coefficients can be viewed as individual local frequency components located at some sub-bands.

  Zeng *et al.* [64] proposed to shuffle those coefficients located at the

same frequency, then randomly changed the sign bits of AC coefficients and flipped DC coefficients with respective threshold. This encryption scheme could achieve high-visual degradation because of the encryption was performed on DC coefficients, where most of the plain-image's energy was concentrated. Nonetheless, this method was vulnerable to chosen-plain-text attack and known-plain-text attack since it was a permutation-only based method, replacing the DC coefficients with a fixed value would give an intelligible version of plain-image. Moreover, the shuffle operation resulted in a 20% bitrate increase and decreased the final compression ratio [15].

In [58–62], Au Yeung *et al.* proposed to realize perceptual video encryption by embedding the encryption scheme at the transformation stage during the encoding process. Since transformation is an important step in video encoder and decoder, a significant advantage of this scheme is that it introduces nearly zeros extra computations. They generated a number of new orthogonal transforms with similar coding efficiency to DCT-II by introducing extra rotation angles into butterflies of DCT-II's flow-graph structure, then applied these new transforms alternatively for blocks' transformation, according to a predefined secret key, to realize partial video encryption. In [62], they first suggested a simple design to generate new orthogonal transforms by selecting four different sets of rotation angles to be embedded into 4-point DCT's flow-graph structure. Then in order to enlarge the differences among these transforms which could offer a strong robustness against possible attackers, they achieved the maximum difference in [60] by allowing rotation angles 0 or π. These newly produced transforms had exactly the same coding efficiency as DCT since they were the sign-flipped version of the origi-

nal DCT. Next, in [59], Au Yeung *et al.* extended the 4×4 transform based encryption framework into 8×8 case to realize encryption for H.264 and MPEG-4 standard, which allowed more room to do butterfiles' sign-flipping and thus to embed the encryption. This encryption scheme could also be adopted for JPEG image encryption. Three different encryption algorithms with different transforms had been proposed in this work, and *Algorithm-3* had been proved to have the best performance between encryption and compression performance. Their scheme was format-compliant to the compressor, could control the encryption degree without affecting the compression performance of the underlying video compression standard. However, the security of this approach was limited because it only modified the transformation stage for encryption, and the 8×8 blocks' independently encryption technique possessed no diffusion property, which may be vulnerable to differential attack.

- *Encryption at quantization stage.* Encryption is mainly achieved by modifying value of entries in the quantization table [70–72] or encrypting quantized DC and AC coefficients [73–75]. In [73], Tang proposed a method to realize image encryption by splitting all quantized DC coefficients in 8×8 blocks into two parts: the four lowest bits were set to the value of DC coefficient, while the highest four bits were set to the last AC coefficient, and then all quantized coefficients in one 8×8 block were shuffled. Coefficients shuffle changed the energy distribution in the frequency domain, and rendered the cipher-image incomprehensible. However, at the same time, coefficients shuffle degraded the effect of zigzag scan and introduced about 40% loss in the compression efficiency [76].

  Later, Lu *et al.* [74] performed image encryption by scrambling quantized DC and AC coefficients. To maintain high compression ratio, zeros

were usually not scrambled. However, this method was vulnerable to the non-zero-counting attack (NZCA) developed in [75] by Li *et al.*. To solve this problem, a full inter-block shuffle JPEG image encryption was proposed in [75], in which quantized coefficients within the same sub-bands were shuffled. Although the proposed method was secure with respect to NZCA, it suffered from severe bitstream size increment [70]. In [70] and [71], Ong *et al.* and Qian *et al.*, respectively, proposed methods for encryption at JPEG's quantization stage by changing the magnitude of entries in the quantization table. However, only changing the quantization table did not offer adequate security against various attacks, it often accompanied with other stages' encryption to offer a higher protection power, and the changed entries' magnitudes had an impact on the final compression ratio.

- *Encryption at zigzag scan stage.* At this stage, the encryption is generally realized by changing the zigzag-scanning order [73, 77–79] and/or encrypting the scanned coefficients [80]. The first MPEG encryption scheme was proposed by Tang [73], known as "zigzag permutation algorithm". The basic idea of this scheme was to use a random permutation list to replace the original zigzag order in mapping the 8×8 DCT block to a 1×64 vector. Shin *et al.* [77] then proposed a very similar cryptosystem for MPEG video by encrypting the sign bits of DC coefficients and using a random scan instead of a zigzag. However, several shortcomings of the zigzag permutation algorithm were identified in [81]. First, the zigzag permutation algorithm is vulnerable to a cipher-text-only attack as this attack utilizes the non-zero AC coefficients located at the upper left corner of the considered 8×8 image block.

  Based on the statistical analysis which count the number of non-zero

DC and AC coefficients in all 8×8 blocks, Qiao and Nahrstedt [82] gave some examples to show how well images could be approximated. Second, permutations are known to be vulnerable against known-plain-text attack, since attackers can retrieve the permutation list by comparing the original and permuted coefficients, under the condition that a certain plain-image is known to the attacker. Additionally, assuming the decoder is available to an attacker, Uehara *et al.* also showed that the analogous weakness of zigzag permutation algorithm against a chosen-plain-text attack [83]. In [78] and [79], S. S. Maniccam *et al.* and S. Jha, respectively, proposed joint encryption and compression schemes based on the SCAN language, which is a formal language based on a two-dimensional spatial accessing methodology that is capable of representing and creating a great variety of 2-D array scanning paths from a small set of primitive ones. This type of encryption schemes had high security level with good confusion and diffusion properties, but its main drawback was that the high computational overhead would cause delay in the compression/encryption procedure and was not so compression friendliness.

Another image encryption method after zigzag scan was developed by Ji *et al.* through encrypting the selected coefficients ranged in $[-64, -5) \bigcup (5, 63]$ and shuffling the positions of identifiers so to distinguish each quantized 8×8 block [80]. The proposed method could obtain high compression performance and robust security simultaneously, however, the shuffle operation on identifiers might lead to decoding errors when applying JPEG's entropy decoding, giving rise to a JPEG non-compliant problem.

- *Encryption at entropy coding stage.* Various techniques, such as multi-

ple Huffman tables for DC/AC coefficients' entropy coding [68,84,85] or bitstream modification and encryption [13,71,86–90], can be applied for entropy coding stage encryption. In [68,84,85], Wu *et al.* proposed a multimedia encryption scheme in the entropy coding stage by using multiple Huffman coding tables alternating in a secret order. Four Huffman tables were developed, which could be further used to derive thousands of different tables based on a technique called Huffman tree mutation. This encryption scheme could achieve high-visual degradation with no impact on the compression performance, because the encryption did not affect the probability distribution of symbols. Nonetheless, the technique was not format-compliant to the corresponding compression standard, since the decoder needed to decrypt the Huffman tables that was used for encryption in order to do decompression. Moreover, Zhou *et al.* [91] and G. Jakimoski *et al.* [92] had pointed out that Wu's encryption scheme [84] was vulnerable to chosen-plain-text/known-plain-text attacks.

A general selection encryption approach for variable-length codes (VLC) was proposed in [13] by Wen *et al.*. They first selected VLC codewords corresponding to information fields critical to reconstructing the image/video quality, then concatenating the codewords for encrypting, was followed by assigning each codeword in the VLC table with a fixed length code index, and then encrypting the indices using DES. Finally, they mapped the encrypted concatenated indices back to a different but valid VLC. This technique had acceptable security level and was fully compliant to any compressor that adopted a VLC entropy coder. However, it compromised the compression efficiency, because some short VLC codewords, which were the most suitable/probable, might be replaced by long codewords.

Next, in 2002, another JPEG compliant encryption method was proposed by M. Van Droogenbroeck *et al.* through only encrypting those appended bits that were used to fully specify the magnitudes and signs of non-zero coefficients in the Huffman coding stage [86]. For this scheme, the encryption was independent of the Huffman coder and had no effect on the final compression efficiency. However, it used static encryption parameters, hence it could not control the quality of the final cipher-images. In [71], Qian *et al.* performed JPEG image encryption by encrypting all appended bits of the Huffman codes with a stream cipher, with the Huffman codes kept unchanged. After encryption, the file size was preserved and the format was compliant to common JPEG decoders. However, Qian *et al.* in [87] pointed out that encryption algorithm in [71] was not adequately secure. This was due to the fact that an adversary could use all the Huffman codes and then set all appended bits to zero, he/she could reconstruct a contour of the plain-image. To improve the security, Qian *et al.* proposed to first randomly select only part of JPEG's bitstream segments, then used a stream cipher to encrypt all appended bits of the DC and AC Huffman codes inside these selected segments to obtain a smaller sized JPEG cipher-image [71]. Thus, with limited Huffman codes, adversary could not recover the contour of the plain-image.

2.3.3 Post-compression encryption algorithms

Encryption algorithms from post-compression class perform encryption after compression, and decompression after decryption. This class of algorithms is generally compression-friendly, they merely produce small overheads to send the encryption key or some information about encryption. No modifications are required for encryption and decryption at the encoder and decoder sides. It

was suggested by D. Socek *et al.* in [93] that post-compression cryptosystems were inherently non-format compliant, because these encryption techniques operated directly on the compressed data stream which might destroy the format information of these data. Conventional encryption algorithms, such as DES and RC4, often encrypt data in block/byte/bit unit through XOR operation, regardless of the relationships existed among blocks/bytes/bits, thus, completely destroy the format information of the data stream. In [16], Xu *et al.* proposed to encrypt the JPEG compressed data stream using the variable modular encryption method, which was based on spacing mapping. This scheme was compression-friendly, because it mapped each codeword to another codeword with the same code length. This scheme may confront the format non-compliance problem, since adopting its mapping strategy, the number of elements in each 8×8 block may exceed 64. In [94], Zhang *et al.* produced two chaotic sequences and applied them for scrambling the JPEG image in a basic unit of an 8×8 data block. The proposed encryption scheme kept the size of the encrypted image file or the decrypted image file to be the same as the original file. However, in [37], D. Ponnain *et al.* pointed out that the permutation could only introduce confusion in the plain image, and mere confusion cannot hide the statistical properties. Moreover, the histogram of the encrypted image would be the same as that of the original image, and is vulnerable to the statistical attack. Futhermore, the low diffusion property would make the whole cryptosystem unable to resist the differential attack appropriately.

2.4 Summary

In this chapter, we have briefly described the concept of cryptography and cryptanalysis, and introduced the work flow of JPEG compressor. By exploring positions for embedding encryption techniques into the compression process,

three types of image cryptosystems can be obtained: pre-compression encryption algorithm, in-compression encryption algorithm, and post-compression encryption algorithm. Some existing work of these three types encryption algorithms are explained, together with their advantages and disadvantages. Generally, pre-compression encryption algorithms are inherently format compliant and usually not compression-friendly; in-compression encryption algorithms may have the possibility of impacting format compliance and compression efficiency; post-compression encryption algorithms are usually non-format compliant.

In this thesis, in order to minimise the computational overhead, we will focus on analysing encryption and compression simultaneously, which means to perform encryption operations during the compression procedure. Four different encryption strategies are proposed in this thesis to solve the two common problems existed in in-compression encryption algorithms, non-format compliant problem and decreased compression efficiency problem. In the following four chapters, we will present these four encryption schemes in details.

Chapter 3

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

In this chapter, we propose a novel joint image compression and encryption scheme based on JPEG standard. The encryption is realized at JPEG's transformation stage. Instead of only using 8×8 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, we can control the quality of encrypted images. The encryption scheme is further enhanced by performing block permutation before the entropy encoding stage. Extensive experiments are conducted to show the good security and compression performance of our encryption schemes.

We organize this chapter as follows. In Section 3.1, we introduce the method to generate new order-8 orthogonal transforms which will be used alternatively for transformation according to the encryption key. Section 3.2

describes the encryption/decryption algorithm, evaluate and compare its performance with JPEG and Au Yeung's *Algorithm-3* [59], together with the quality control realization of encrypted images. Security analysis of our proposed scheme is given in Section 3.3. Finally, a summary of the chapter will be presented in Section 3.4.

3.1 Generation of new order-8 orthogonal transforms

In JPEG, 8×8 type-II DCT is used for blocks' transformation, because it can provide an efficient energy compaction performance with separability property when high correlation is existed among inter-pixels [95] – which is a general phenomenon in most natural pictures. In [96], a fast DCT-II algorithm was proposed in the form of matrix computations 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 angles of π to different butterflies. Hence, we first introduce the underlying fast computational algorithm for DCT-II, then explain how we generate new transforms based on it.

3.1.1 One fast computational algorithm for DCT-II

The fast DCT computational algorithm in [96] is based upon matrix decomposition. In general, an order-N type-II DCT matrix can be written into the following recursive form:

$$\left[C_N^{II}\right] = [P_N] \begin{bmatrix} P_{N/2}^t C_{N/2}^{II} & 0 \\ 0 & R_{N/2} \end{bmatrix} [B_N], \tag{3.1}$$

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

where $[P_N]$ is 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 Figure 3.1, and its fast computational formula can be described as:

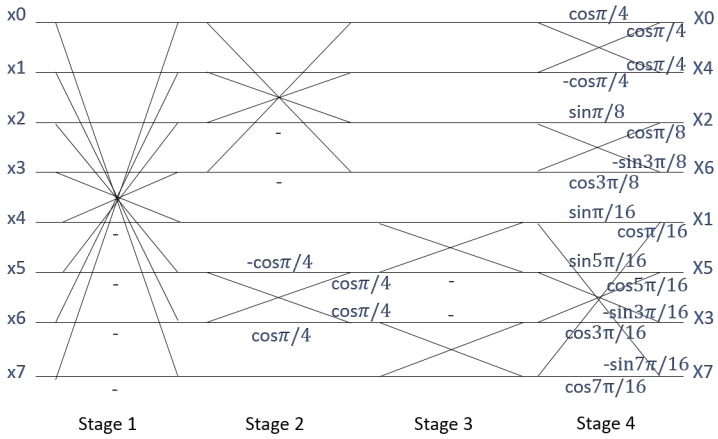


Figure 3.1: Flow graph of 8-point (1-D) DCT

$$\left[C_8^{II}\right] = [P_8] \begin{bmatrix} P_4^t C_4^{II} & 0 \\ 0 & R_4 \end{bmatrix} [B_8], \tag{3.2}$$

$$\begin{aligned}
\left[C_4^{II}\right] &= [P_4]\begin{bmatrix} P_2^t C_2^{II} & 0 \\ 0 & R_2 \end{bmatrix}[B_4]\,, \\
[P_4] &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
[R_2] &= [M1] = \begin{bmatrix} \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \\ -\sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} \end{bmatrix}, \\
[P_2] &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \left[C_2^{II}\right] = \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},
\end{aligned}$$

$$\begin{aligned}
[P_8] &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \\
[R_4] &= [M1]\,[M2]\,[M3]\,,
\end{aligned}$$

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 Figure 3.1.

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

Computation of $[M2]$ is equivalent to the butterfly denoted in the bottom half of Stage-3 in Figure 3.1.

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

Computation of $[M3]$ is equivalent to the butterfly denoted in the bottom half of Stage-2 in Figure 3.1.

3.1.2 Generation of new orthogonal transforms

Considering the compression performance, the coding efficiency of the newly generated transforms should either be exactly the same to what can be achieved

by DCT or just fall slightly. Therefore, we do not introduce angle rotation to butterflies in Stage-1 and Stage-2, because this will cause significant changes in the transformed coefficients, which will eventually affect the overall coding efficiency.

In our transform generation method, using the extra rotation angles 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 Figure 3.1, which can be described using the following mathematical formula:

$$\left[C_8^{II}\right] = [P_8]\,[T_1] \begin{bmatrix} P_4^t C_4^{II} & 0 \\ 0 & R_4 \end{bmatrix} [B_8]\,, \tag{3.3}$$

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 version 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 Figure 3.1, which can be described using the following mathematical formula:

$$\begin{aligned} \left[C_8^{II}\right] &= [P_8]\,[T_1] \begin{bmatrix} P_4^t C_4^{II} & 0 \\ 0 & R_4' \end{bmatrix} [B_8]\,, \\ \left[R_4'\right] &= [M1]\,[T_2]\,[M2]\,[M3]\,, \end{aligned} \tag{3.4}$$

where $[T_2] = diag(\cos\theta_5, \cos\theta_5, \cos\theta_6, \cos\theta_6)$, $\theta_i = 0$ or π $(i = 5 \text{ or } 6)$, 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.

3.2 Joint image compression and encryption based on alternating transforms

3.2.1 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 is considered to be one of the most commonly-used random key generators [97]. The encryption algorithm is stated as follows:

Encryption Algorithm-1-ch3

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 procedures;

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

In *Algorithm-1-ch3*, 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 inverse-transformation.

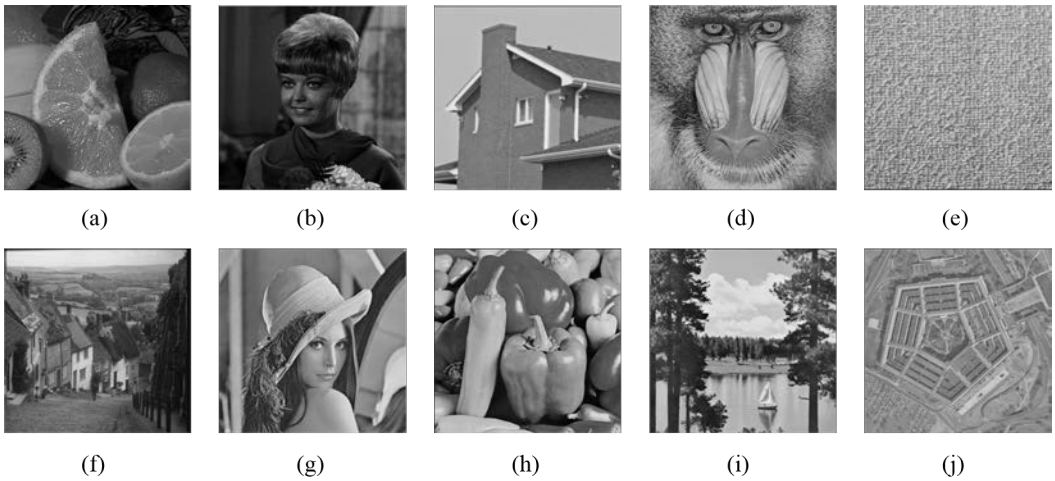
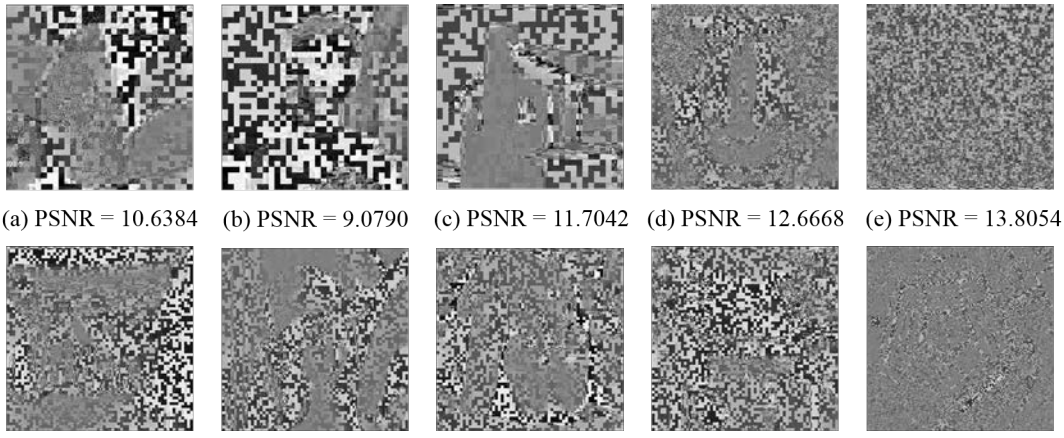


Figure 3.2: Ten test images. (a) Fruits 256×256. (b) Girl 256×256. (c) House 256×256. (d) Baboon 512×512. (e) Raffia 512×512. (f) Goldhill 512×512. (g) Lena 512×512. (h) Peppers 512×512. (i) Sailboat 512×512. (j) Pentagon 1024×1024.

To evaluate the performance of *Algorithm-1-ch3*, we use ten images with different sizes as shown in Figure 3.2 for testing. We encrypt them using *Algorithm-1-ch3* with QF being 20, and their corresponding cipher-images are shown in Figure 3.3, together with the PSNR values. Here, QF = 20 is just used as an example. We assume that the standard inverse discrete cosine transform (IDCT) is always used for decryption when the encryption key is unknown. Performance comparison between our proposed encryption algorithm and Au Yeung's method [59] is shown in Figure 3.4, in which 'Baboon' image is used. In Figure 3.4(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 visual protection ability of our encryption algorithm. When the encryption key is known, the PSNR value, which reflects the coding efficiency of transforms, of

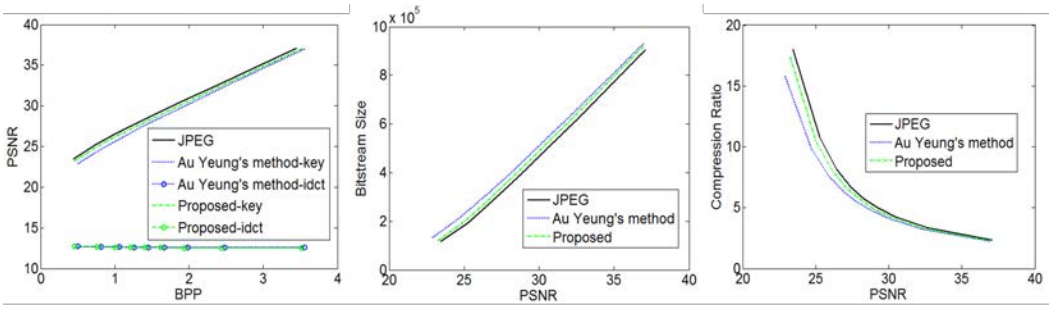
Algorithm-1-ch3 is closer to that of JPEG standard than that of the reference paper's algorithm, illustrating that our transform sets' coding efficiency is better than the transform sets in [59]. For the PSNR-BS (bitstream size) relationship and PSNR-CR (compression ratio) relationship shown in Figure 3.4(b) and Figure 3.4(c), compared with JPEG standard, our algorithm also has better compression performance and smaller bitstream size than [59]'s encryption method.



(a) PSNR = 10.6384 (b) PSNR = 9.0790 (c) PSNR = 11.7042 (d) PSNR = 12.6668 (e) PSNR = 13.8054

(f) PSNR = 9.6051 (g) PSNR = 11.5716 (h) PSNR = 10.4836 (i) PSNR = 8.8049 (j) PSNR = 16.2664

Figure 3.3: Ten cipher-images encrypted by *Algorithm-1-ch3*.



(a) (b) (c)

Figure 3.4: Performance comparison among different encryption algorithms for image 'Baboon': (a) BPP-PSNR relationship, (b) PSNR-BS relationship, (c) PSNR-CR relationship.

3.2.2 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 3.1 has listed the different PSNR performances of various selections of θ_i ($i = 1, 2, 3,$ or 4) for encryption. Three images are used for testing with QF to be 20. From Table 3.1, 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 Figure 3.5, we present the visual results of the three representative selections in Table 3.1 of these three images. These results clearly show 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.

3.2.3 Improved encryption scheme

In Figure 3.3, we can observe that the encrypted images under *Algorithm-1-ch3* still reveal some information about the original images. 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 entropy coding, we first disrupt the

Table 3.1: PSNR [dB] performance of various selection of θ_i for encryption

Angle change \ Image	Fruits	Baboon	Pentagon
None (a)	28.9957	25.2637	29.1710
Stage-3 (b)	23.4720	19.5740	23.8324
Stage-3 & θ_1	10.4439	12.8794	16.3910
Stage-3 & θ_2	22.0732	17.9612	22.4716
Stage-3 & θ_3	22.0667	18.2691	22.1970
Stage-3 & θ_4	22.8635	19.1528	23.5414
Stage-3 & $\theta_1\theta_2$	10.5368	12.5814	16.2867
Stage-3 & $\theta_1\theta_3$	10.5436	12.6679	16.3214
Stage-3 & $\theta_1\theta_4$	10.5496	12.7027	16.3710
Stage-3 & $\theta_2\theta_3$ (c)	21.5909	17.2796	21.3672
Stage-3 & $\theta_2\theta_4$	22.2737	17.7995	22.3619
Stage-3 & $\theta_3\theta_4$	22.1025	18.0595	21.9361
Stage-3 & $\theta_1\theta_2\theta_3$	10.7060	12.7146	16.3575
Stage-3 & $\theta_1\theta_2\theta_4$	10.7099	12.7128	16.3447
Stage-3 & $\theta_1\theta_3\theta_4$	10.6989	12.7490	16.3603
Stage-3 & $\theta_2\theta_3\theta_4$	21.5132	17.0787	21.2525
Stage-3 & $\theta_1\theta_2\theta_3\theta_4$ (d)	10.6384	12.6668	16.2664

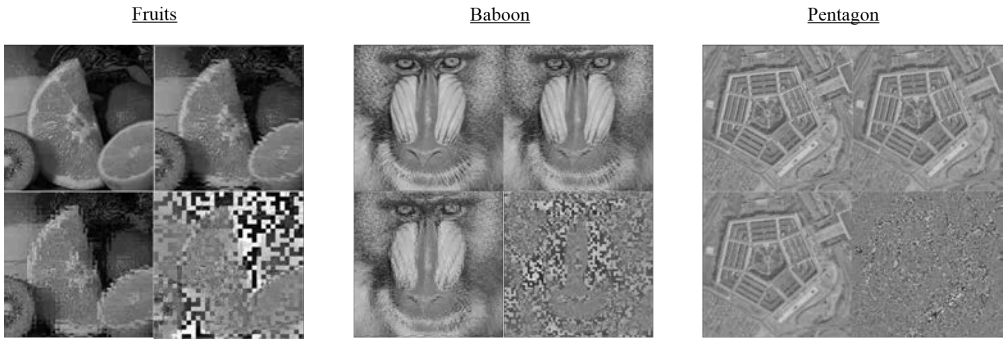


Figure 3.5: Encrypted images with different rotation angles change (left-top: Case (a) in Table 3.1; right-top: Case (b) in Table 3.1; left-bottom: Case (c) in Table 3.1; right-bottom: Case (d) in Table 3.1.

order of these 8×8 blocks, according to the encryption key. The permutation algorithm we select is Fisher-Yates Shuffle because of its simple implementation, which is used for generating a random permutation of a finite linear array [98]. 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

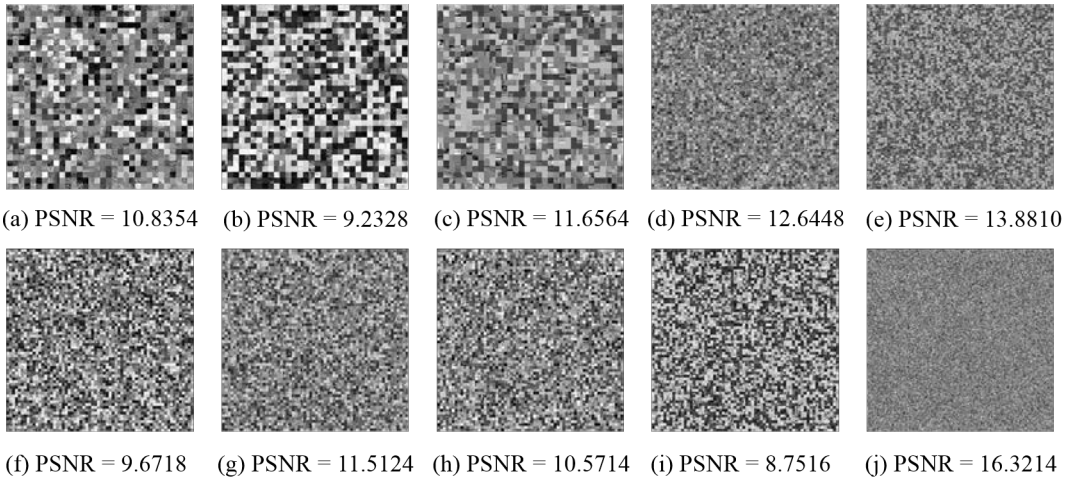
When applying this shuffle algorithm in our encryption scheme, S is the original order of all 8×8 blocks, n is the number of 8×8 block, the random integer in each loop is obtained from the RC4 generated pseudo-random bit-stream, which can be described as following:

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-2-ch3*. The corresponding encrypted ten images are shown in Figure 3.6. It is obvious that the encrypted images under *Algorithm-2-ch3* are more chaotic than images encrypted by *Algorithm-1-ch3*. 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-ch3* can be maintained.



(a) PSNR = 10.8354 (b) PSNR = 9.2328 (c) PSNR = 11.6564 (d) PSNR = 12.6448 (e) PSNR = 13.8810

(f) PSNR = 9.6718 (g) PSNR = 11.5124 (h) PSNR = 10.5714 (i) PSNR = 8.7516 (j) PSNR = 16.3214

Figure 3.6: Ten cipher-images encrypted by *Algorithm-2-ch3*.

3.3 Security analysis

3.3.1 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. To make brute-force attack infeasible, cryptosystem's key space should be large enough [99]. 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-ch3* 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.

3.3.2 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 [100]. 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 [101]:

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:

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

Pseudo-codes 1 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-codes 2 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

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 3.2. Moreover, in Figure 3.7, we have taken 'Baboon' as an example to present the cipher images with slightly different encryption key and the difference image of *Algorithm-1-ch3* and *Algorithm-2-ch3*. The low correlation coefficient and chaotic difference images prove the fulfilment of condition 1 of

key sensitivity analysis for our proposed encryption scheme.

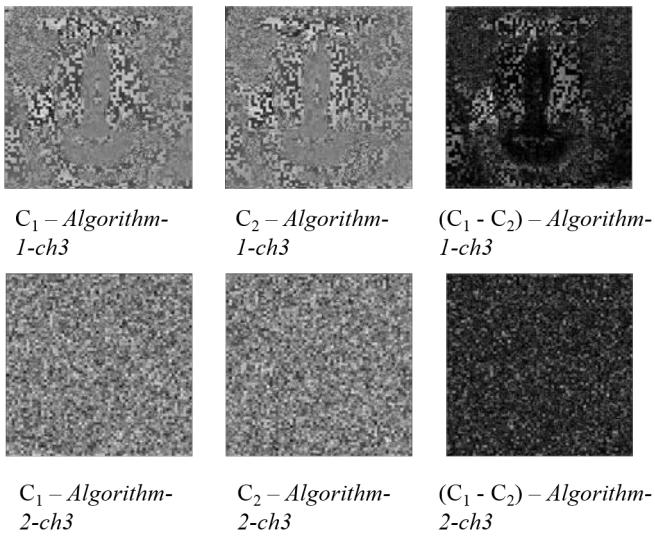


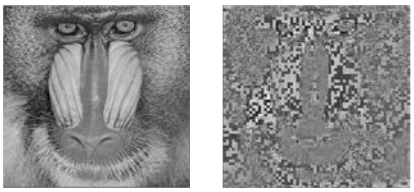
Figure 3.7: Key sensitivity analysis for encryption process — Encrypted 'Baboon' image with slightly modified encryption keys and their difference images.

Table 3.2: Correlation coefficient for image encrypted with original key and slightly different key

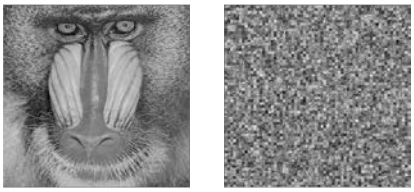
Algorithm / Image	*Algorithm-1-ch3*	*Algorithm-2-ch3*
Fruits	-0.0365	-0.0139
Girl	0.0361	0.0300
House	0.0155	-0.0053
Baboon	-0.0010	-0.0173
Raffia	0.0037	0.0042
Goldhill	0.0001	-0.0201
Lena	0.0059	-0.0150
Peppers	-0.0090	0.0196
Sailboat	0.0012	0.0180
Pentagon	0.0013	0.0120

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, 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 'Baboon' under *Algorithm-1-ch3* and *Algorithm-2-ch3* obtained with slightly different decryption keys are shown in Figure 3.8. 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.



Algorithm-1-ch3



Algorithm-2-ch3

Figure 3.8: Key sensitivity analysis for decryption process: left image decrypted with right encryption key, right image decrypted with slightly modified decryption key.

3.3.3 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 [14]. 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 the cipher media content. Correlation-based replacement is similar to direct replacement with the difference that some of the encrypted data is replaced

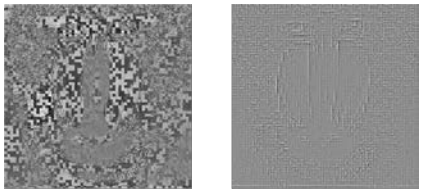
by unencrypted data.

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 'Baboon' images under our proposed two algorithms (*Algorithm-1-ch3*, *Algorithm-2-ch3*) and of Au Yeung's *Algorithm-3* [59] are shown in Figure 3.9. We can observe that encryption with block permutation have better defense ability against the direct replacement attack than encryption without block permutation and Au Yeung's *Algorithm-3* [59], which both reveal some outline information about the plain Baboon image after the direct replacement attack operation.

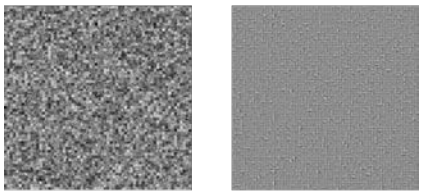
3.3.4 Statistical model-based attack

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 [14]. 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.

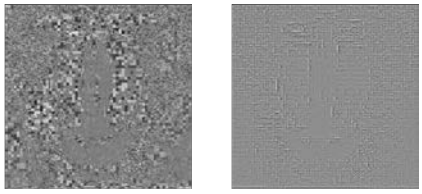
Apart from the PSNR measure, attackers also can study the relationship between plain image and cipher image without knowing the encryption key to decrypt the cipher image. Generally, histograms and correlation diagrams of



(a) *Algorithm-1-ch3*



(b) *Algorithm-2-ch3*



(c) Au Yeung's method

Figure 3.9: Direct replacement attack analysis for different encryption algorithms (left: encrypt image, right: decrypt image).

the original image and the encrypted image are two ways to indicate the degree of relationship between the plain image and cipher image [66, 95, 102, 103]. Histogram analysis shows the distribution of pixels in the image by counting the number of each pixel's brightness [104]. The block permutation operation does not change pixel values, thus the histograms of encrypted image with/without permutation remain the same. In Figure 3.10, we present histograms of the original 'Baboon' image and its corresponding cipher image under *Algorithm-2-ch3*. It can be seen that histogram of the encrypted 'Baboon' image has little statistical similarity to that of the plain image. But the histogram does not show the uniform 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 neighbour-

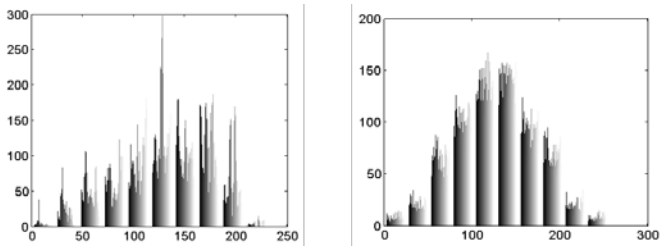


Figure 3.10: Histogram charts of 'Baboon' image (left: plain image, right: *Algorithm-2-ch3* encrypted image).

hood of diagonal pixels from the original image and the encrypted image. Then 1000 pairs of two adjacent pixels are randomly selected, the correlation diagram is plotted based on the value of each pixel and its diagonal neighbours. The corresponding diagram of 'Baboon' image encrypted by *Algorithm-2-ch3* is shown in Figure 3.11. 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 as significant 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.

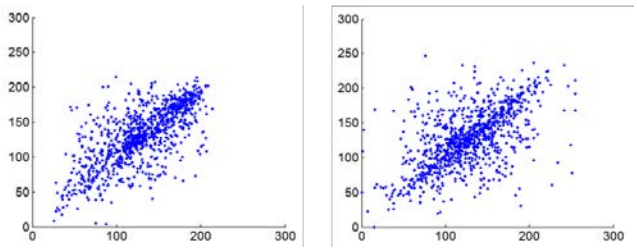


Figure 3.11: Diagonal correlation charts of 'Baboon' image (left: plain image, right: *Algorithm-2-ch3* encrypted image).

3.4 Summary

In this chapter, a new joint image compression and encryption scheme is proposed to realize image encryption at JPEG's transformation stage. The pro-

posed 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 the 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 sacrificing the JPEG compression. 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.

Currently, the proposed scheme's security level is not so strong, because we only consider JPEG image encryption during transformation. In the next chapter, we will explore the possibility of combining our current transformation-stage-only encryption scheme with JPEG other stages' encryption, such as entropy coding stage encryption, so as to enlarge the encryption space by one dimension.

Chapter 4

A content-adaptive joint image compression and encryption scheme

For joint image compression and encryption schemes, encryption power and compression efficiency are commonly two contradictory things. In this chapter, we propose a new joint image compression and encryption scheme based on lossy JPEG standard, which aims at improving the encryption security, on the premise of maintaining the efficiency of JPEG compression. The proposed scheme is image-content-adaptive, since we generate the secret encryption key from the plain-image using BLAKE2 hash algorithm. Three encryption operations are included in the 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 decode 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 the proposed scheme

is JPEG friendly, has good confusion and diffusion properties. Detailed security analysis is also given to illustrate the scheme's robustness against various cryptanalysis strategies.

The major contributions of this chapter can be summarized as follows.

- The mechanism of producing encryption key from the plain-image can greatly improve the cryptosystem's diffusion property and hence enhance the scheme's protection performance against the differential attack, which is often lacked in many joint compression-encryption schemes, like [59, 71, 84].

- New orthogonal order-8 transforms are developed with similar coding efficiency to DCT by introducing rotation angles into order-8 DCT-II's flow graph, and they are utilized for 8×8 blocks' transformation alternatively controlled by the secret key.

- A data hiding technique is proposed by upgrading some AC coefficients' category without breaking the format information of the final encrypted bitstream.

- The whole scheme is JPEG compression friendly, can achieve a good balance between the encryption power and the compression efficiency.

We organize this chapter as follows. In Section 4.1, we explain the implementation details of our encryption operations, and the corresponding encryption and decryption algorithms are also given. Experimental results are presented in Section 4.2. Section 4.3 analyses the cryptosystem's security level against various types of attacks. Summary of the chapter is given in Section 4.4.

4.1 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 4.1.1 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 4.1.2, and Section 4.1.3 explains our data embedding/extracting strategy for encrypting AC coefficients, which is controlled by the predefined embedding key, denoted as Key_2. The encryption algorithm and decryption algorithm are presented in Section 4.1.4.

In [105], Zhang *et al.* pointed out that some joint image compression and encryption cryptosystems contained security flaws, such as methods in [106, 107]. 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, such as 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. Hence, in our scheme, we generate the secret encryption key based on the plain-image through the BLAKE2 hash function. The reason why BLAKE2 is adopted is because of its simple implementation and high security. Any minor change occurred at any positions of the plain-image will produce totally different encryption keys, since BLAKE2 is very sensitive to the changes made in its input. Therefore, the proposed scheme can resist the chosen-plain-text attack.

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, denoted as Key_1 and named encryption key, 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 follows

$$K_{n+1} = H(K_n)(n = 0, 1, 2, ...) \tag{4.1}$$

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}.

4.1.1 New orthogonal transforms transformation

To generate new orthogonal transforms, we also introduce rotation angles into order-8 DCT-II's flow-graph structure like our first encryption method, described in Chapter 3. The rotation angels and embedding positions, however, are different from those taken in the first encryption scheme. Here, we allow more rotation angles, rather than only 0 and π, and we only introduce different rotations angles into the four butterflies at Stage-4 in Figure 3.1, which can be represented by modifying Equation (3.2) using the following formula:

$$[C_8^{II}] = [P_8][T_1] \begin{bmatrix} P_4^t C_4^{II} & 0 \\ 0 & R_4 \end{bmatrix} [B_8], \tag{4.2}$$

where $[T_1] = diag(\cos\alpha_1, \cos\alpha_1, \cos\alpha_2, \cos\alpha_2, \cos\alpha_3, \cos\alpha_4, \cos\alpha_4, \cos\alpha_3)$, $\alpha_1 = 0$ or π, $\alpha_i \in (0, \frac{\pi}{3}, \frac{2\pi}{3}, \pi)$ $(i = 2, 3, \text{ or } 4)$, 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 significant impact on the overall coding efficiency of the new transform. Using Equation (4.2), we can totally produce 128 different orthogonal transforms, and we name the new transform set as *MTS1* (modified transform set 1).

To evaluate the coding efficiency of *MTS1*, we compare its performance with DCT, transforms used in [59]'s *Algorithm-3*, and transforms used in our first encryption method through a statistical model. For the *Algorithm-3* in [59], 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 Figure 3.1, and all eight rows used the same transform matrix for transformation, decided by the secret key. For the column's transformation, a total of 256 orthogonal transforms were generated by introducing rotation angles (0 or π) into eight butterflies of Figure 3.1 (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 [108] and transform efficiency [109] 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:

$$\begin{aligned} G_{TC} &= \frac{\frac{1}{n}\sum_{i=0}^{n-1} s_{ii}}{\left(\prod_{i=0}^{n-1} s_{ii}\right)^{1/n}} \\ \text{Efficiency } \eta &= \frac{\sum_{i=0}^{n-1} |s_{ii}|}{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} |s_{ij}|} \times 100\% \end{aligned}$$

The average transform coding gain G_{TC} and average transform efficiency η versus different correlation coefficients ρ for DCT, transforms in *MTS1*, transforms in Au Yeung's method [59], and transforms used in *Algorithm-1-ch3* are shown in Table 4.1 and Table 4.2, respectively. It is clear that the coding efficiency of our transform set is better than that of Au Yeung's method and *Algorithm-1-ch3*, and very close to DCT[1].

[1]In our experiment, we use 32 digits precision to display numbers, and the minor differences in G_{TC} and η values of *MTS1* and DCT can be seen.

Table 4.1: 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
MTS1	2.4162	3.0386	4.2424	7.6312
Au Yeung's method	2.2092	2.7553	3.8160	6.8119
Algorithm-1-ch3	2.3065	2.8852	4.0064	7.1666

Table 4.2: 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
MTS1	0.8497	0.8695	0.8984	0.9399
Au Yeung's method	0.7748	0.8026	0.8440	0.9056
Algorithm-1-ch3	0.8105	0.8344	0.8698	0.9219

When using *MTS1* 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.

4.1.2 DC coefficients encryption

To introduce more disorder to the encrypted image, after JPEG quantization process, we propose to permute all order-8 quantized blocks, and the permutation vector is generated from the Fisher-Yates shuffle [98] controlled by K_{enc-1}, which has been described in Section 3.2.3. After performing the block permutation operation for all 8×8 quantized blocks, we change the DC coefficient in each 8×8 permuted block using Equation (4.3), to further improve

the diffusion and confusion properties.

$$dc_i = dc_i \oplus dc_{i-1} \oplus dc_{i-2} \oplus \cdots \oplus dc_1, \tag{4.3}$$

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, Equation (4.4) can be used:

$$dc_j = dc_j \oplus dc_{j-1}, \tag{4.4}$$

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

4.1.3 AC coefficients encryption

To minimise 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 these coefficients.

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 coefficient positions into an array $Index$ for the

control of data embedding. The whole operation is described in Figure 4.1.

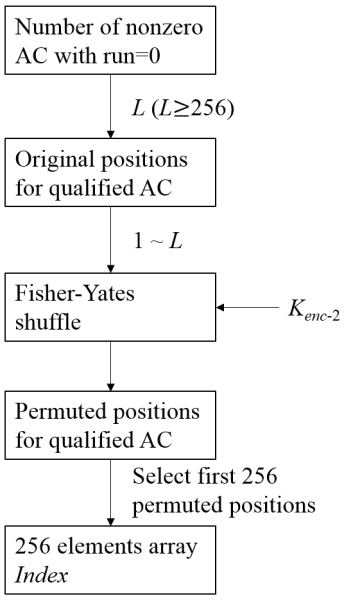


Figure 4.1: Positions selection for data embedding.

For data embedding, 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, we do not change the Huffman coding result of this coefficient. If yes, increase the coefficient's category size by 1, then look up the Huffman table for JPEG 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 Figure 4.2 to better explain our 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 perform the procedures as shown in Figure 4.1, so to obtain the 256 embedding positions array $Index$. Given the entropy encoded bits of one AC coefficient, the decoder first checks whether

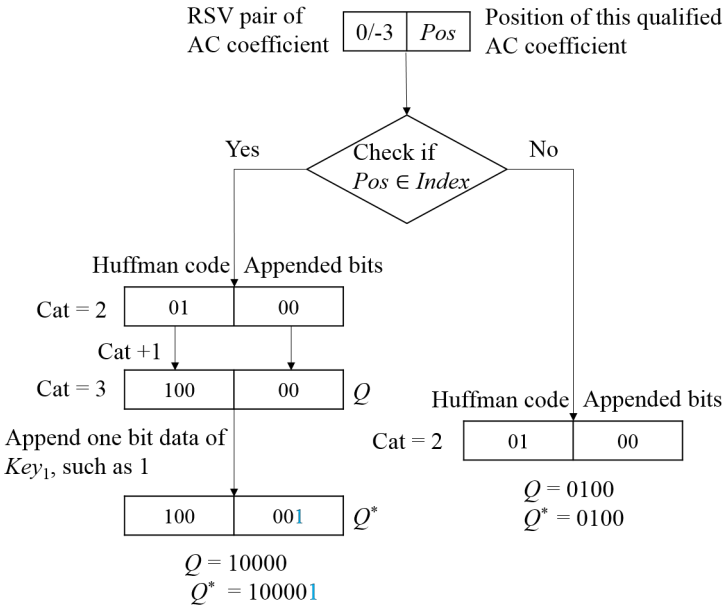


Figure 4.2: Example of data embedding strategy.

its run-length is 0 by 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 size 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 Figure 4.3, which is the reverse procedure of Figure 4.2.

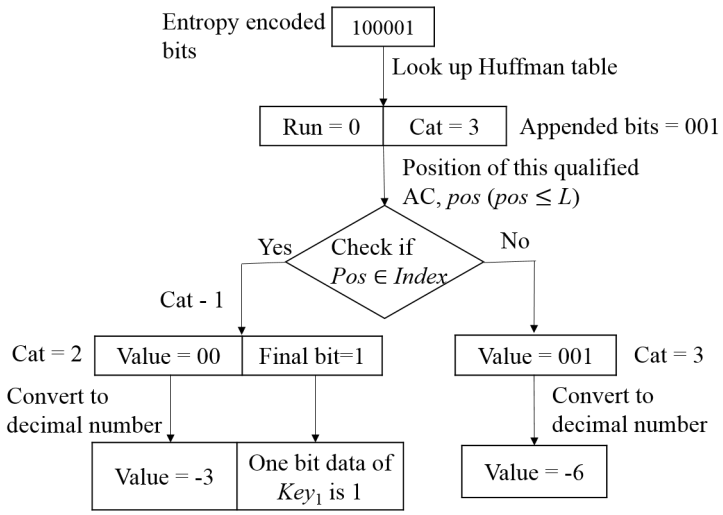


Figure 4.3: Example of data extracting strategy.

It is noted that our data embedding scheme cannot apply for those AC coefficients having a size of 10 bits, which may appear when QF is greater than or equal to 93, otherwise, the problem of having more than 64 coefficients in a 8×8 block will be incurred due to the category upgrading process and hence leads to the non-format compliant problem. To solve the problem, we use different AC coefficients' selection strategies under different QF values. When QF<93, all AC coefficients in a block will be considered for data embedding. For the case when QF≥93, we first zigzag scan the QF scaled quantization table, and compute the last occurrence position of value '1', denoted as $LastPos$, since after DCT, the range of AC coefficients is (-1024, 1024), and only step size of '1' in quantization table can produce 10 bits quantized AC coefficients. Then, we select qualified AC coefficients starting from each 8×8 quantized block's $(LastPos + 1)^{th}$ coefficient based on the zigzag order. The $LastPos$ values corresponding to QF value from 93 to 100 are 6, 6, 10, 13, 21, 25, 54, and 64, respectively. Nonetheless, 10 bits quantized AC coefficients have a very low probability to appear at each 8×8 block's bottom-right positions, because elements' values of JPEG's quantization table are increasing under the zigzag scan order. Therefore, in our experiment, we set $LastPos = 30$ when QF = 99 and QF = 100 and no problem is found in all 1857 images, which will be introduced in Section 4.2.

4.1.4 Encryption and decryption algorithms

The realization procedures of our proposed cryptosystem are presented in Figure 4.4, and the encryption and decryption algorithms can be described as follows:

Encryption Algorithm-1-ch4

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 *MTS1* for *all* rows' transformation in the 1^{st} dimension;

Step-2.3: Use the following 7×8 bits to select one transform from *MTS1* 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 Equation (4.3);

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-1-ch4

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 Equation (4.4) 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 *MTS1* for *all*

rows' inverse-transformation in the 1^{st} dimension;

Step-4.3: Use the following 7×8 bits to select one transform from *MTS1* for *each* column's inverse-transformation in the 2^{nd} dimension;

Step-5: Repeat *Step-4* until all 8×8 blocks are inverse-transformed, then we can obtain the decrypted image.

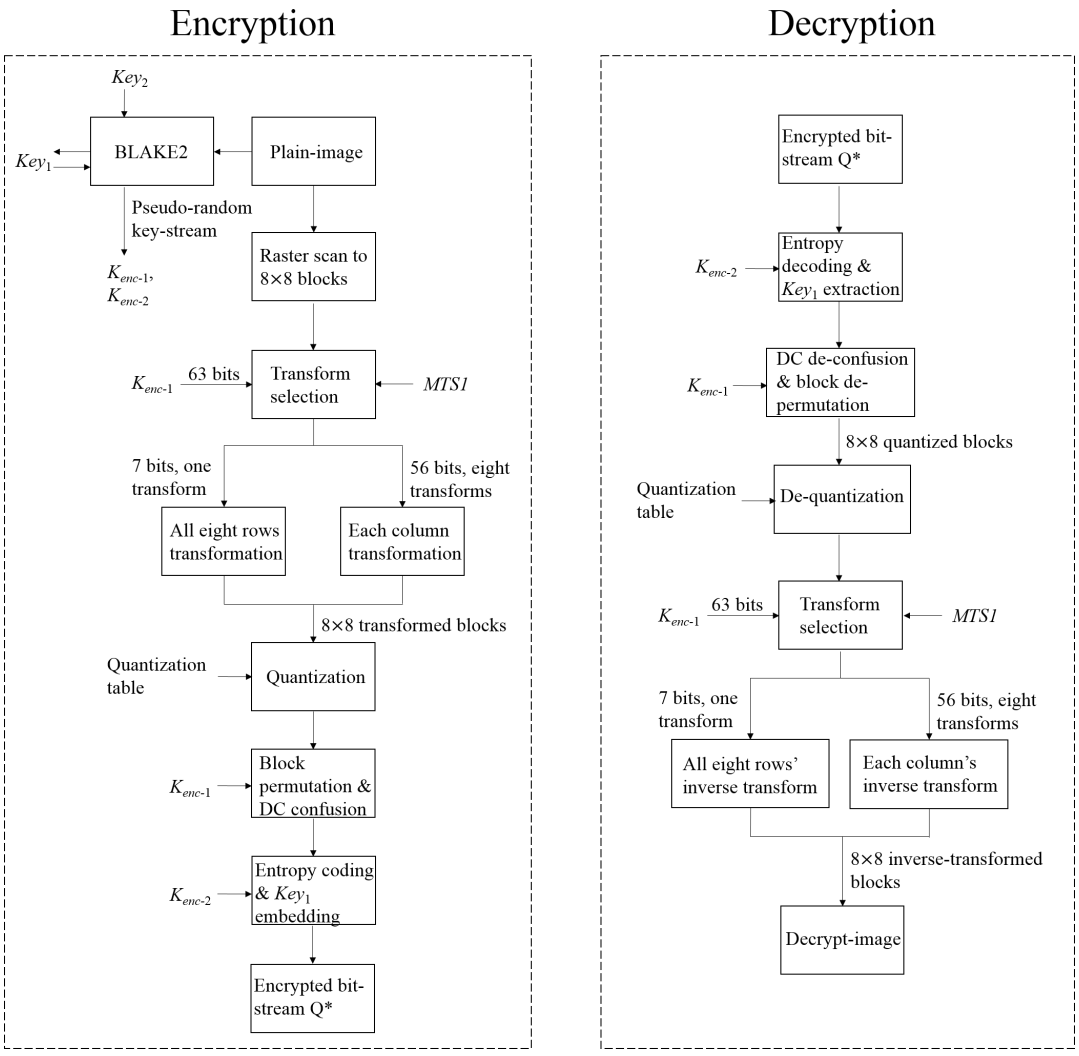


Figure 4.4: Encryption and decryption procedures of our scheme.

4.2 Experimental results

In this section, the perceptual security of our proposed scheme and the compression performance on the encrypted images are evaluated experimentally. Perceptual security refers to the perceptual distortion of cipher-image with respect to the plain-image, and the PSNR criterion is adopted to measure it. Three popular image databases are used for testing: USC-SIPI image database (totally 306 images, available on the website "http://sipi.usc.edu/database"), UCID image database [110] (totally 1338 images), and JAFFE facial expression database [111] (totally 213 images). Here, we choose ten gray images, shown in Fig. 4.5, as experiment samples. We encrypt these ten images using the proposed scheme with QF being 20, and their corresponding cipher-images are shown in Figure 4.6. It can be observed that these cipher-images are totally randomised and no outline information about their plain-images can be seen.

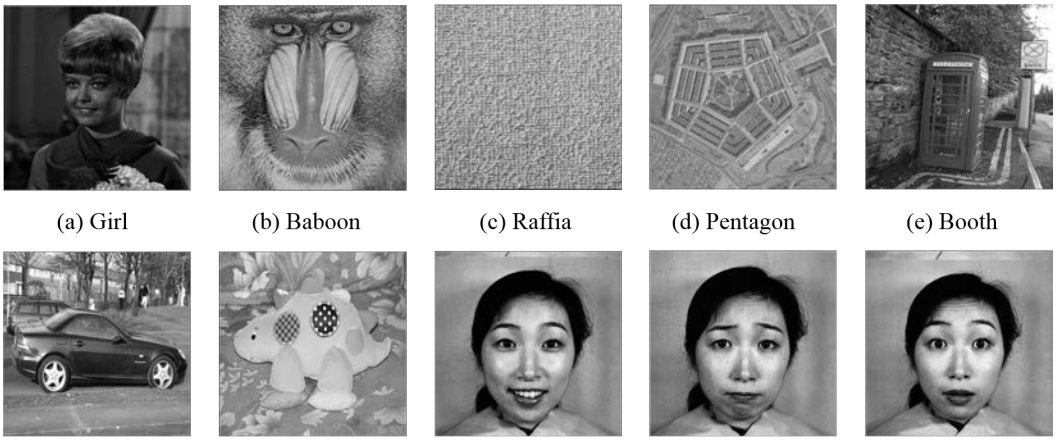
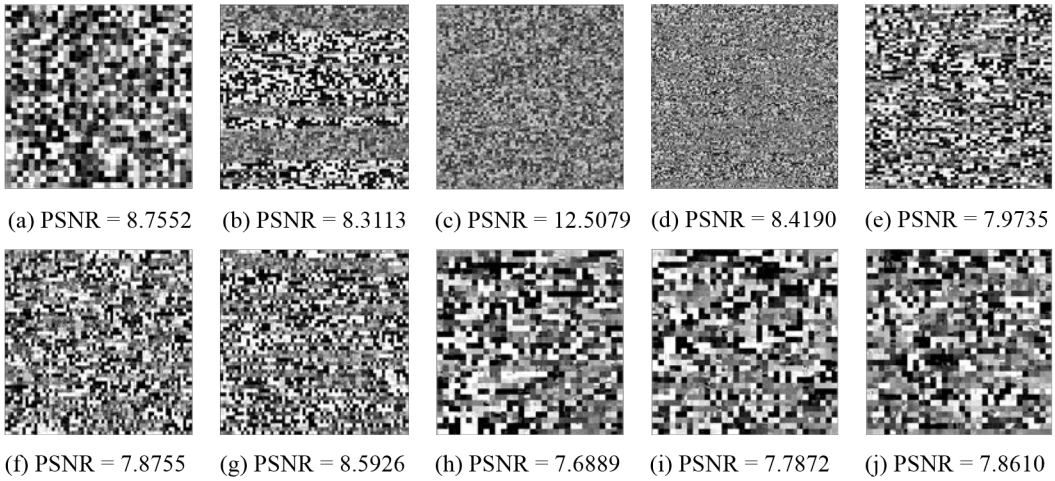


Figure 4.5: Ten test images from different image databases.

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



(a) PSNR = 8.7552 (b) PSNR = 8.3113 (c) PSNR = 12.5079 (d) PSNR = 8.4190 (e) PSNR = 7.9735

(f) PSNR = 7.8755 (g) PSNR = 8.5926 (h) PSNR = 7.6889 (i) PSNR = 7.7872 (j) PSNR = 7.8610

Figure 4.6: Ten cipher-images encrypted by *Algorithm-1-ch4*.

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 BPP and PSNR values for various images with different QF values (QF = 20, QF = 40, QF = 60, and QF = 80) are presented in Tables 4.3 through 4.6, 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 from our proposed method under different QF values are a little larger than results of JPEG, while their PSNR values are nearly the same, confirming that our proposed encryption scheme is compression friendly to JPEG.

To better illustrate *Algorithm-1-ch4*'s performance, we use all images of three image databases for encryption and decryption with QF ranging from 15 to 95. Their average BPP-PSNR curves are plotted in Figure 4.7, and they are compared with JPEG, Au Yeung's *Algorithm-3* [59] and the encryption algorithm proposed in Chapter 3 (*Algorithm-1-ch3*). From the BPP-PSNR curves, we can see that when the encryption key is not known and only IDCT is used for different cryptosystems' decompression and decryption, the average PSNR values of our encryption scheme have the largest drop, which means a higher perceptual security. Additionally, when the key is available to the

Table 4.3: Comparison of compression performance when QF = 20

Image	Proposed		JPEG	
	BPP	PSNR	BPP	PSNR
Girl	0.4213	32.8159	0.3647	32.8161
Baboon	0.7778	25.2637	0.7440	25.2637
Raffia	0.7873	28.8030	0.7590	28.8030
Pentagon	0.4933	29.1710	0.4591	29.1710
Booth	0.6901	27.3212	0.6470	27.3206
Car	0.6130	28.5283	0.5695	28.5283
Toy	0.4638	31.0767	0.4175	31.0767
Joy	0.4049	30.8929	0.3639	30.8930
Sadness	0.3982	31.1351	0.3558	31.1337
Surprise	0.4024	30.7613	0.3615	30.7616

Table 4.4: Comparison of compression performance when QF = 40

Image	Proposed		JPEG	
	BPP	PSNR	BPP	PSNR
Girl	0.6234	35.0051	0.5544	35.0026
Baboon	1.2290	27.3780	1.1883	27.3779
Raffia	1.1754	31.2211	1.1417	31.2211
Pentagon	0.7992	30.8560	0.7700	30.8560
Booth	1.0780	29.3362	1.0276	29.3362
Car	0.9552	30.6180	0.9028	30.6179
Toy	0.7375	33.1940	0.6829	33.1936
Joy	0.6474	32.3727	0.5993	32.3737
Sadness	0.6336	32.5747	0.5812	32.5732
Surprise	0.6530	32.2661	0.6021	32.2647

decoder, the average PSNR values obtained by our current scheme are the closest to JPEG, 0.7 - 1.5 dB higher than Au Yeung's PSNR value and 0.4 - 0.9 dB higher than *Algorithm-1-ch3*'s PSNR value, respectively, under the same BPP value. This also indicates the coding efficiency of our new transform set is better than that of Au Yeung's method [59] and *Algorithm-1-ch3*, matching the results of coding efficiency comparison using the statistical model in Section 4.1.1. In Table 4.7, we also compare the encryption efficiency for different sizes

Table 4.5: Comparison of compression performance when QF = 60

Image	Proposed		JPEG	
	BPP	PSNR	BPP	PSNR
Girl	0.8161	36.3435	0.7374	36.3416
Baboon	1.6215	29.1476	1.5832	29.1476
Raffia	1.5042	32.7025	1.4606	32.7026
Pentagon	1.0979	32.0463	1.0607	32.0463
Booth	1.4216	30.8725	1.3579	30.8723
Car	1.2708	32.1201	1.2099	32.1191
Toy	0.9812	34.6526	0.9275	34.6529
Joy	0.9020	33.4013	0.8454	33.4018
Sadness	0.8878	33.5739	0.8258	33.5730
Surprise	0.9170	33.3429	0.8540	33.3407

Table 4.6: Comparison of compression performance when QF = 80

Image	Proposed		JPEG	
	BPP	PSNR	BPP	PSNR
Girl	1.2245	38.5559	1.1329	38.5569
Baboon	2.4414	32.5956	2.3802	32.5955
Raffia	2.1832	35.4463	2.1329	35.4462
Pentagon	1.7358	34.3140	1.6898	34.3139
Booth	2.1266	34.0064	2.0559	34.0060
Car	1.9011	34.8975	1.8300	34.8978
Toy	1.4846	37.1310	1.4212	37.1307
Joy	1.4730	35.2256	1.4040	35.2248
Sadness	1.4486	35.4101	1.3782	35.4102
Surprise	1.4841	35.2482	1.4134	35.2466

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.

[2]Computation cost of BLAKE2 has been included.

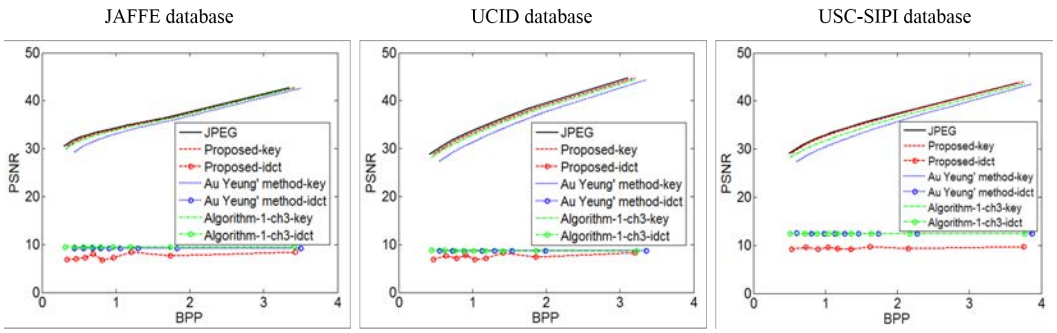


Figure 4.7: Comparison of BPP-PSNR curves for different encryption schemes.

Table 4.7: 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

4.3 Security analysis

In the previous section, we have proved that the proposed content-adaptive image 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.

4.3.1 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. 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 $\left(\frac{L-255}{256}\right)^{256}$. In our scheme, after JPEG quantization process, on average each 8×8 block should have at least one qualified AC coefficient, for a plain-image of size 256×256, totally 1024 8×8 blocks, the minimum value for L would be 1024, $\left(\frac{L-255}{256}\right)^{256} \approx 3^{256} > 2^{256}$, which illustrates that breaking our cryptosystem through guessing the 256 embedding positions is still very hard.

4.3.2 Jigsaw puzzle solver attack

In [53], T. Chuman *et al.* proposed the use of jigsaw puzzle solvers as one of attack methods on encryption. They claimed 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 [53], and our scheme can be classified into the Type 1 puzzle because only block scrambling operation is performed 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 four small images (256×256 color 'Girl' image, 384×512 color 'Booth' image, 512×384 color 'Car' image and 512×384 color 'Toy' image) for testing, and the jigsaw piece size is 8×8 pixels.

The reason why we do not choose database JAFFE's 256×256 facial expressions images is because they are gray images. The same three measures in [53] (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 [53], the assembling results for Type 1 puzzles with piece size being 8×8 were not given. We first use the only 8×8 blocks scrambling encryption method and our proposed method to encrypt the four images, then apply [53]'s jigsaw puzzle solver to recover these cipher-images, corresponding experimental results are shown in Table 4.8. Based on the low values of three measures, we believe 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 4.8: Jigsaw puzzle solver results

Image	Scramble only			Proposed		
	Dc	Nc	Lc	Dc	Nc	Lc
Girl	0.001	0.301	0.089	0.000	0.000	0.001
Booth	0.000	0.272	0.042	0.000	0.000	0.000
Car	0.000	0.153	0.008	0.000	0.000	0.001
Toy	0.000	0.267	0.052	0.000	0.000	0.001

4.3.3 Replacement attack and sketch attack

Here we only analyse the direct replacement attack on the proposed cryptosystem, since we encrypt the whole image together, do not rank the information contained in the plain-image. The direct replacement attack proposed in [84] is evaluated on our scheme by assigning all DC coefficients to 128 and all AC coefficients to positive. We use our proposed method to encrypt three different images, 'Baboon', 'Car', and 'Joy', the encrypt images and attacking results are shown in Figure 4.8, from which we can observe that no information of the

plain-images can be distinguished in our scheme.

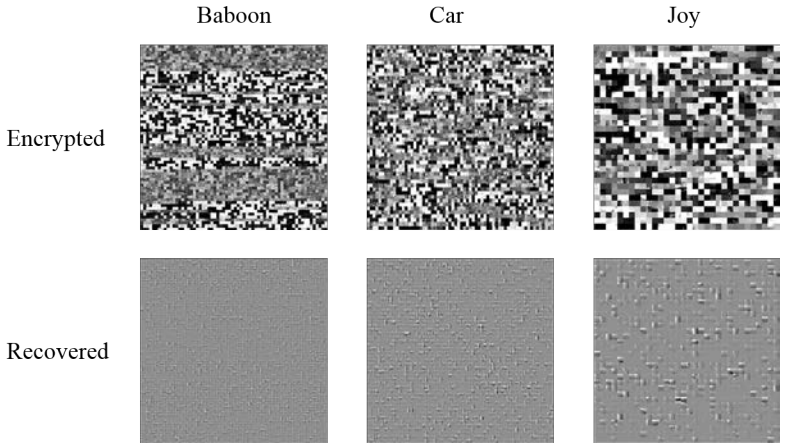


Figure 4.8: Direct replacement attack results.

Sketch attack generally tries to sketch the outline of the original image/video frame directly from the encrypted image/video. In [112], several conventional sketch attacks, Position of Last Non-zero Coefficient (PLZ) [113], Sum of Absolute AC Coefficients (SAC) [114], and DC Error Category [115] for JPEG images had been introduced by K. Minemura *et al.*, and a new sketch attack based on macroblock bitstream size was proposed for sketching H.264/AVC video frames [112]. Two methods to resist this type of attack were reported, one used macroblocks permutation while ensuring format-compliance, while the other diffused the visual information of a region into other regions. As indicated in [112], the main disadvantage when taking these two encryption methods was the reduced compression efficiency. In our proposed cryptosystem, we adopt these two methods (8×8 blocks' permutation and DC coefficients confusion) while keeping format compliance. The final compression efficiency of JPEG can be maintained, therefore we believe our encryption method is robust against sketch attack.

4.3.4 Key sensitivity analysis

In our proposed scheme, we use BLAKE2 hash function to produce encryption key Key_1 and the two pseudo-random key-streams. 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 a significant variation in the produced pseudo-random key-streams, and alter all the following encryption operations. To verify the key sensitivity with respect to the encryption process, 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 Figure 4.9, and their difference image is also presented. The chaotic difference image of these two cipher-images proves key sensitivity analysis of our encryption process.

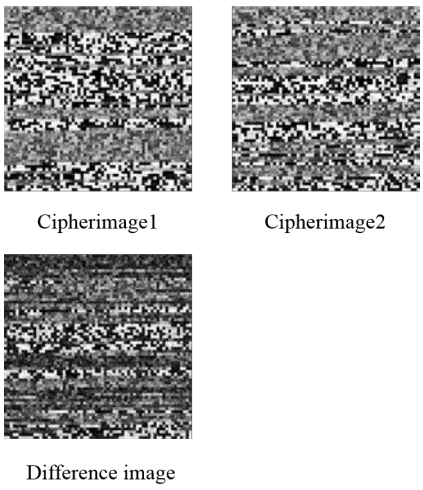


Figure 4.9: Key sensitivity analysis for encryption process.

To evaluate the key sensitivity with respect to our decryption 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 Figure 4.10. It is clear that even when there is a small variation in

Key_2, correct decryption cannot be realized under our encryption/decryption scheme.

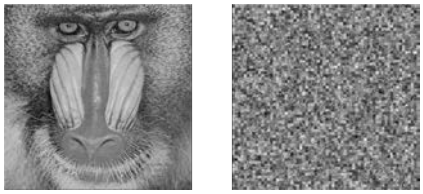


Figure 4.10: Key sensitivity analysis for decryption process: left image decrypted by the original key, right image decrypted by the modified key.

4.3.5 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, an attacker tries to deduce a statistical relationship between them [101]. A cryptosystem is considered robust 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) [116], are widely used for checking the robustness of an image encryption scheme against differential attack [117]. NPCR denotes the change rate of the number of pixels in the cipher-image while one pixel in plain-image is changed [117, 118]. The higher the value of NPCR, the more secure is the encryption scheme. UACI measures the average intensity difference between two cipher-images [118]. To get higher security, the value of UACI should be close to 33% [116]. 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 [119]:

$$D(i,j) = \begin{cases} 0, if\ C^1(i,j) = C^2(i,j), \\ 1, if\ C^1(i,j) \neq C^2(i,j). \end{cases} \tag{4.5}$$

$$\text{NPCR} : N(C^1, C^2) = \frac{\sum_{i,j} D(i,j)}{H \times W} \times 100\%, \tag{4.6}$$

$$\text{UACI} : U(C^1, C^2) = \frac{\sum_{i,j} \frac{|C^1(i,j) - C^2(i,j)|}{255}}{H \times W} \times 100\%, \tag{4.7}$$

NPCR and UACI analyses have been tested on various images by increasing its first pixel by 1. In Table 4.9, the NPCR and UACI values for images encrypted by our algorithm, Au Yeung's *Algorithm-3* [59], and *Algorithm-1-ch3* are listed. From the table, we can observe that our encryption scheme has a certain degree of robustness against differential attack. While for Au Yeung's algorithm and *Algorithm-1-ch3*, the NPCR and UACI values are all almost zero, indicating their low diffusion property.

Table 4.9: Results of differential attack tests on various images

Image	Au Yeung's method [59]		*Algorithm-1-ch3*		Proposed	
	NPCR%	UACI%	NPCR%	UACI%	NPCR%	UACI%
Girl	0	0	0	0	96.62	39.96
Baboon	0	0	0	0	97.80	39.48
Raffia	2.4033e-04	3.7399e-06	0	0	99.49	23.68
Pentagon	0	0	0	0	99.43	21.97
Booth	0	0	0	0	97.97	39.27
Car	0	0	0	0	97.62	39.74
Toy	0	0	0	0	97.48	40.13
Joy	0	0	0	0	97.23	39.74
Sadness	0	0	0	0	97.63	41.44
Surprise	7.0190e-04	4.0690e-06	0	0	97.18	41.29

4.3.6 Statistical model-based attack

We also use histogram and correlation charts, like Chapter 3, to show the relationship existed between plain-image and cipher-image.The histogram of plain 'Baboon' and encrypted image under the current proposed scheme are shown in Figure 4.11. It is clear that the histogram of encrypted image has low statistical similarity to that of the plain image. However, it does not show the uniform distribution property, which indicates a more secure level. This is because our encryption scheme is based on the 8×8 block unit and the correlation among pixels in the encrypted images cannot be destroyed completely. For the correlation chart, we initially identify the neighbourhood of diagonal

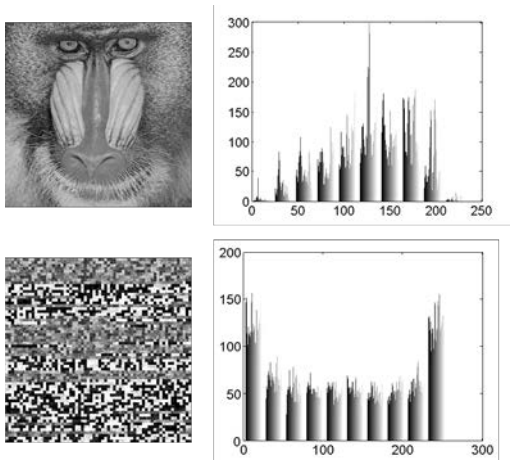


Figure 4.11: Histogram charts of plain 'Baboon' image and encrypted image.

pixels from the plain-image and cipher-image. Then 1000 pairs of two diagonal adjacent pixels are randomly selected. The correlation diagram is plotted 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 Figure 4.12. We can observe that the linear property, which reflects the correlation degree between pixels, shown in cipher-image is slightly less than the chart shown for the plain-image. However, it cannot be totally removed, as a result of encryption being performed in 8×8 blocks unit, similar

to JPEG.

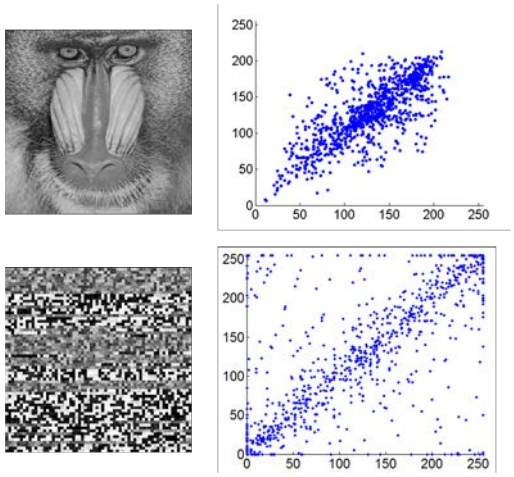


Figure 4.12: Correlation charts of plain 'Baboon' image and encrypted image.

4.4 Summary

In this chapter, we have proposed a new joint image compression and encryption scheme, in which the encryption key is dependent on the plain-image. The whole scheme has better diffusion property than the method proposed in Chapter 3, 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. The encryption operations include three parts: new orthogonal 8×8 transforms transformation, DC coefficients encryption, and AC encryption. Extensive experiments and security analysis are conducted to show the good compression and encryption performance of our proposed encryption scheme.

Although the protection ability and compression performance of the proposed scheme are better than the encryption method proposed in Chapter 3, it cannot achieve perfect correlation removal effect and diffusion property, because the whole encryption work is still realized in 8×8 block unit. Hence, in

our next step, we will try to do transformation in different block sizes, so as to enhance the cryptosystem's correlation removal ability.

Chapter 5

Joint image compression and encryption schemes based on 16×16 DCT

Joint image encryption and compression schemes have shown their great potential values in protecting compressed images, which are the most common form of Internet images. To achieve high security, a trade-off between encryption power and compression ability needs to be considered. To satisfy different encryption/compression requirements, in this chapter, we propose two new joint encryption and compression schemes by incorporating scrambling techniques in different intermediate stages of the JPEG process, where one scheme emphasizes compression performance, another highlights security. In order to enhance the security ability against the chosen-plain-text attack, in both schemes, we use adaptive key for each plain-image's encryption/decryption, and the key is dependent on the plain-image. The initial encryption operation we make in the two proposed schemes is to replace JPEG's original 8×8 DCT with 16×16 DCT, which means that we raster scan the plain-image into non-overlapping 16×16 blocks, and the following encryption operations are all

based on it. In the first encryption scheme, after applying order-16 DCT for all 16×16 blocks, we perform coefficients distribution, block permutation, and DC coefficients confusion for encryption, all controlled by the adaptive key. As for our second encryption scheme, to further improve security, we add the run/size and value (RSV) pairs' shuffling operation in the entropy coding stage, on the basis of the first scheme. Performance evaluations on these two encryption schemes using various criteria are conducted, and the results have shown that the first scheme has better compression efficiency, while the second scheme has better defense ability against the differential attack and the statistical attack.

We organize this chapter as follows. In Section 5.1, we introduce how we produce the plain-image-dependent secret key and the pseudo-random key-stream. Section 5.2 explains the implementation details of our first 16×16 DCT-based encryption method. Section 5.3 presents the added RSV shuffling operation in the second encryption scheme. Detailed performance evaluations on the compression efficiency and the encryption power of the two cryptosystems are given in Section 5.4. Finally, summary of the chapter is given in Section 5.5.

5.1 Key processor

To introduce diffusion property into the encryption process with respect to the plain-image, in both our encryption schemes, we first use the BLAKE-256 hashing function [120] taking the plain-image as input to output a 256-bit random hash value, denoted as σ. The reason why we choose BLAKE-256 is because of its fast speed, security, and simplicity. Then we use Chen's chaotic system [121] to produce three pseudo-random key sequences, which can be described as follows:

$$\begin{cases} \dot{x} = a(y - x), \\ \dot{y} = (c - a)x - xz + cy, \\ \dot{z} = xy - bz. \end{cases} \tag{5.1}$$

where a, b and c are parameters which determine system's chaotic attractors and bifurcations. When $a = 35$, $b = 3$, $c \in [20, 28.4]$, the system is chaotic as shown in Figure 5.1. The reason why we choose Chen's chaotic system is because of its abundant and complex dynamic behaviours, which are very useful in secure communications [122].

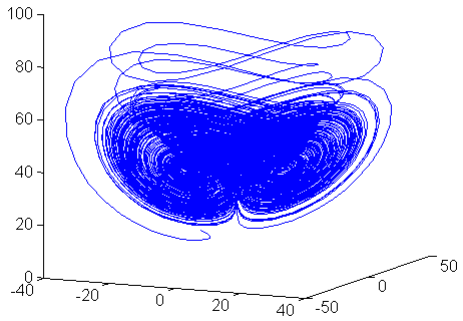


Figure 5.1: Chen's chaotic system.

We convert the 256-bit hash value σ to 8-bit decimal form: $\sigma = k_1 k_2 \dots k_{32}$, where each k_i ranged in $[0, 255]$. Then these 32 decimal numbers are used to modify the initial states of Chen's chaotic system as follows,

$$\begin{aligned} x_0 &= x_0' + \frac{(k_1 \oplus k_2 \oplus \dots k_{11})}{256}, \\ y_0 &= y_0' + \frac{(k_{12} \oplus k_{13} \oplus \dots k_{22})}{256}, \\ z_0 &= z_0' + \frac{(k_{23} \oplus k_{24} \oplus \dots k_{32})}{256}, \end{aligned} \tag{5.2}$$

where x_0', y_0', and z_0' are the given initial states, which are served as secret parameters. We iterate Chen's chaotic system for 10000 times with the new

initial values x_0, y_0, and z_0, and for each iteration, three values x_i, y_i, and z_i can be obtained, $i = 1, 2, ..., 10000$. We pre-process these values using the following formula,

$$X = dec2bin(((abs(x_i) - floor(abs(x_i))) \times 10^{14}) \mod (H \times W)),$$
$$Y = dec2bin(((abs(y_i) - floor(abs(y_i))) \times 10^{14}) \mod (H \times W)),$$
$$Z = dec2bin(((abs(z_i) - floor(abs(z_i))) \times 10^{14}) \mod (H \times W)),$$

where function $dec2bin(x)$ converts decimal number x into binary number, $abs(x)$ returns the absolute value of x, $floor(x)$ rounds the element of x to the nearest integers less than or equal to x. H and W represent the height and width of the plain-image in pixels. Therefore, in our proposed two schemes, the encryption key has two parts: 1) the 256-bit hash value σ; 2) the three initial states x'_0, y'_0, and z'_0. The three binary sequences X, Y, and Z are taken as the pseudo-random key-streams to control following encryption operations.

5.2 First encryption scheme

In Section 2.3, we classify JPEG image encryption into three types (pre-compression encryption, in-compression encryption, and post-compression encryption) based on different positions for introducing encryption operations. Here, we use a figure to better distinguish them. In Figure 5.2, there are six different positions that can be utilized for introducing encryption techniques. Pre-compression encryption algorithms are realized at position (1) of Figure 5.2, In-compression encryption algorithms can be implemented at positions (2)-(5) of Figure 5.2, and Post-compression encryption algorithms are realized at position (6) of Figure 5.2.

In our first encryption scheme, we realize encryption at JPEG's trans-

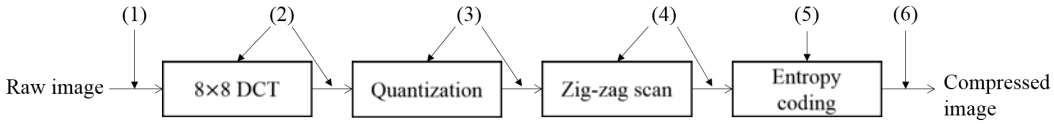


Figure 5.2: Positions for realizing JPEG image encryption.

formation stage and after quantization stage, which correspond to positions (2) and (3) in Figure 5.2. We explain the implementation details of the in-transformation encryption and after-quantization encryption separately.

5.2.1 In-transformation encryption

The modification we made for encryption in this part is to replace the original 8×8 DCT transformation by 16×16 DCT transformation. Hence, we initially raster scan the input plain-image to sequential 16×16 blocks, then apply 16×16 DCT for blocks' transformation. The reason why we do not choose larger block sizes, such as 32×32 or 64×64 block size, for encryption is because if we use order-32 or order-64 DCT for blocks transformation, the quantized DCT coefficients category size may exceed 10 bits, which is the largest category size that JPEGs Huffman table can encode. After transformation, we distribute the 256 coefficients of each 16×16 block (denoted as $B16$) into four 8×8 blocks (represented by a 3-D array $B8$), so that JPEG's following encoding procedures can still be applied to encode these 8×8 blocks. The whole process is controlled by the pseudo-random keystream X. We explain our coefficients distribution procedure using Pseudo-codes 3.

In Pseudo-codes 3, i represents which 8×8 block is participated in this time's coefficients distribution, since there are four 8×8 blocks, 2 bits are needed from key-stream X each time. j indicates the number of coefficients that will be distributed into ith 8×8 block, and we use 3 bits from X to decide this number. The reason why we choose 3 bits is to avoid one 8×8

Pseudo-codes 3 Coefficients Distribution

Input: One 16×16 block $B16$, and keystream X
Output: Four 8×8 blocks ($B8$)

```
CoeIndex ← 1;
B8Number ← 1 × 4 vector with four elements being zero;
while CoeIndex ≤ 256 do
    i ← pick 2 bits from X, and convert to decimal;
    j ← pick 3 bits from X, and convert to decimal;
    if B8Number(i) < 64 then
        Temp ← B8Number(i) + j;
        if (Temp − 64) > 0 then
            j ← j − (Temp − 64);
        end if
        ith 8×8 block of B8 ← pick j elements from B16 in zigzag scan order;
        B8Number(i) ← B8Number(i) + j;
        CoeIndex ← CoeIndex + j;
        Remove the first j elements from B16 in zigzag scan order;
        Remove the first 5 bits from X;
    else
        Remove the first 5 bits from X;
    end if
end while
return B8
```

block possessing too many low frequency coefficients from the 16×16 block, which will result in increased information loss in the encrypted image. In order to count how many coefficients have been possessed by one 8×8 block, we use a 1×4 vector, $B8Number$, to save the coefficients number information of the four 8×8 blocks. When the coefficients distribution is done, all the four elements in $B8Number$ will be 64. During the coefficients distribution process, information denoting which 8×8 blocks possess 16×16 blocks' DC coefficient is also saved as $DCIndex$, and this information will be used in the following after-quantization encryption procedure. After finishing coefficients distribution, we quantize all 8×8 blocks using JPEG's quantization table, and perform the next step, which is encryption.

It is noted that after order-16 DCT's transformation, the range of coefficients is [-2048, 2048) , while JPEG's Huffman tables can only process coefficients ranged in [-1024, 1024). According to the relationship between JPEG's quantization table and QF value, we can find that only when QF≥93 can the quantization table have element '1', which will result in 11 bits quantized DC/AC coefficients. Therefore, to solve the out of range problem of the 16×16 DCT transformed coefficients, we divide all these coefficients by 2 when QF≥93, so that after JPEG's quantization, these coefficients' range becomes [-1024, 1024), and they can be entropy encoded by JPEG's Huffman table correctly.

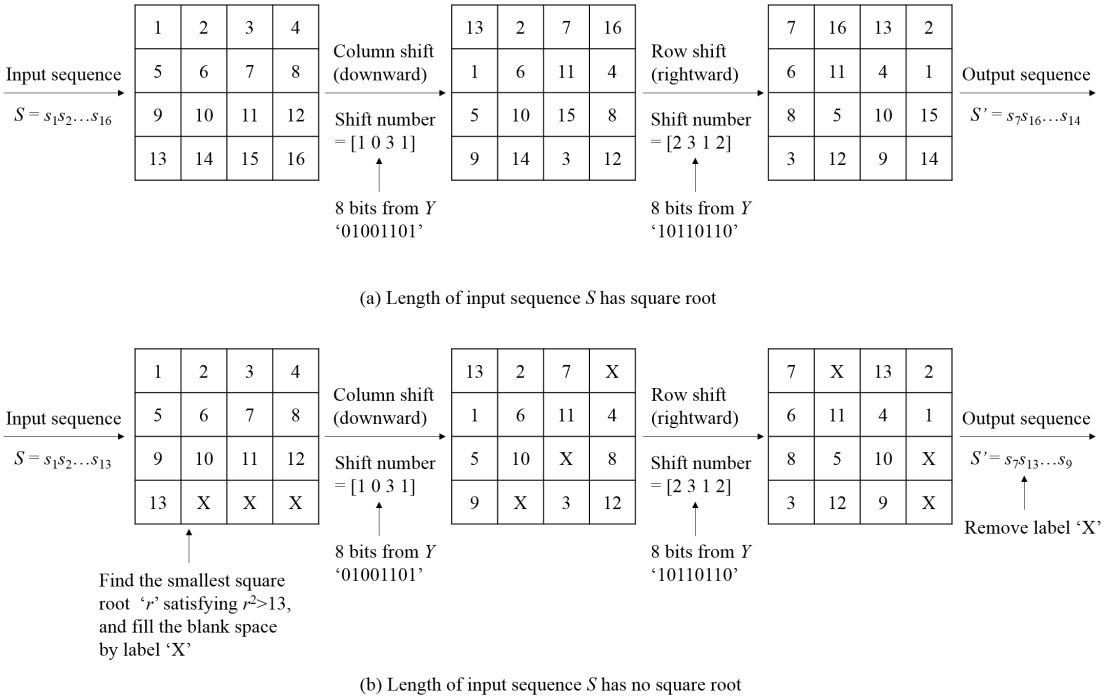
5.2.2 After-quantization encryption

In this part, we permute all quantized 8×8 blocks and confuse the DC coefficients in original 16×16 blocks (denoted by $DCIndex$) through XOR operation using key-stream Y. The reason why we do not perform a confusion on all 8×8 blocks' first coefficient (may be the DC coefficient of 16×16 block or not) is

because after 16×16 DCT transformation and 8×8 blocks' quantization, most coefficients in 8×8 blocks are zero. If we perform a confusion on the first coefficient in all 8×8 blocks, then when inverse 8×8 DCT is performed at the decoder side for decompression, many 8×8 blocks' first coefficient will exceed the range of [0,255], which means that after the normalization process, the cipher-image will contain a significant number of white/black pixels, resulting in the huge loss of image information.

The permutation method we use is the two key-driven cyclical shift [123], where circular shift is operated on each column and each row separately, and the number of shift positions is controlled by Y. The reason why this shuffling method is adopted is because of its simple implementation. For a given input sequence, we first reshape it into a square matrix, then downward circular shift each column's element according to the shift position decided by the first half key-stream of Y. After shifting all columns, rightward circular shift each row's element according to the shift position decided by the second half key-stream of Y. We call the final produced permutation vector as S'. In Figure 5.3, we give two examples to illustrate this permutation method. The two examples explain separately how we produce permutation vector when the length of input sequence S has square root or not. When using this permutation method for our 8×8 blocks' permutation, the input sequence S is the original order of all 8×8 blocks, and the output sequence S' is the permutation vector we need.

After performing block permutation, positions for original 16×16 blocks' DC coefficients ($DCIndex$) are changed. Finding the permuted $DCIndex$ under the permutation vector S', denoted as $DCIndex'$, we perform XOR operation on these permuted DC coefficients according to the order that they appear after block permutation, to further improve the diffusion and confusion properties of our first encryption scheme. Specific computation formula is given



(a) Length of input sequence S has square root

(b) Length of input sequence S has no square root

Figure 5.3: Two examples of circular shift.

as follows:

$$dc_{DCIndex'(i)} = dc_{DCIndex'(i)} \oplus dc_{DCIndex'(i-1)} \oplus \cdots \oplus dc_{DCIndex'(1)}, \tag{5.3}$$

where $dc_{DCIndex'(i)}$ is the DC coefficient of the $DCIndex'(i)$th permuted 8×8 block, which is also the DC coefficient of one 16×16 block, $i = 1, 2, \ldots, H \times W/256$. For recovering these confused DC coefficients, Equation (5.4) can be used:

$$dc_{DCIndex'(j)} = dc_{DCIndex'(j)} \oplus dc_{DCIndex'(j-1)}, \tag{5.4}$$

where j starts from $H \times W/256$, and ends with 2.

Here, we use the first example of Figure 5.3 to illustrate how we confuse the DC coefficients. From the Figure, there are sixteen 8×8 blocks, $s_1 s_2 ... s_{16}$, in which $s_1 s_2 s_3 s_4$, $s_5 s_6 s_7 s_8$, $s_9 s_{10} s_{11} s_{12}$, and $s_{13} s_{14} s_{15} s_{16}$ constitute four 16×16 blocks, respectively. We suppose that after order-16 DCT's transformation and coefficients distribution, s_2, s_5, s_{11} and s_{14} are the four 8×8 blocks who possess

original 16×16 blocks' DC coefficients, then $DCIndex = (2, 5, 11, 14)$. After disrupting these 8×8 blocks using $S' = s_7 s_{16} ... s_{14}$, $DCIndex$ is changed into $DCIndex' = (4, 10, 6, 16)$. We perform a confusion on these DC coefficients according to their order of appearance using Equation (5.3), thus $dc_4 = dc_4$, $dc_6 = dc_6 \oplus dc_4$, $dc_{10} = dc_{10} \oplus dc_6 \oplus dc_4$, and $dc_{16} = dc_{16} \oplus dc_{10} \oplus dc_6 \oplus dc_4$, which are equal to $dc_{s_2} = dc_{s_2}$, $dc_{s_{11}} = dc_{s_{11}} \oplus dc_{s_2}$, $dc_{s_5} = dc_{s_5} \oplus dc_{s_{11}} \oplus dc_{s_2}$, and $dc_{s_{14}} = dc_{s_{14}} \oplus dc_{s_5} \oplus dc_{s_{11}} \oplus dc_{s_2}$.

5.2.3 Encryption and decryption algorithms for the first scheme

Encryption algorithm of our first method mainly contains four parts: a) pseudo-random key-stream generation using BLAKE-256 and Chen's chaotic system; b) 16×16 DCT transformation, and coefficients distribution; c) 8×8 blocks' permutation and DC coefficients confusion; d) entropy encoding.

Encryption Algorithm-1-ch5

Step-1: Take plain-image as the input of BLAKE-256 algorithm to generate a 256-bit hash value σ, use this data to modify the three initial values (x'_0, y'_0, and z'_0) by Equation (5.2), then run Chen's chaotic system for 10000 times to produce three pseudo-random key-streams X, Y, and Z;

Step-2: Level shift the plain-image by subtracting 128 from each pixel, and segment the level shifted image into non-overlapping 16×16 blocks. For each 16×16 image block, do

Step-2.1: Transform the block using 16×16 DCT;

Step-2.2: Distribute the 256 coefficients into four 8×8 blocks controlled by key-stream X, quantize these 8×8 blocks by JPEG's original order-8 quantization table;

Step-3: Repeat *Step-2* until all 8×8 blocks are quantized, permute these blocks using the two key-driven cyclical shift and key-stream Y, and confuse DC coefficients according to Equation (5.3);

Step-4: Perform JPEG's entropy coding procedure for all processed 8×8 blocks, transmit the encrypted bit-stream, and encryption keys (256-bit hash value σ, three initial values x'_0, y'_0, and z'_0) securely to decoder for decryption and decompression.

For decrypting the encrypted bit-stream, if encryption keys are not known, we just follow JPEG's decompression process to obtain the cipher-image, because of the format-compliant property. When keys are available, the decryption algorithm is described as follows:

Decryption Algorithm-1-ch5

Step-1: Use σ to modify x'_0, y'_0, and z'_0 by Equation (5.2), run Chen's chaotic system for 10000 times produce the pseudo-random key-streams X, Y, and Z;

Step-2: Perform entropy decoding procedure of JPEG, and generate the 8×8 blocks' permutation vector S' using the two key-driven cyclical shift and key-stream Y;

Step-3: Recover the confused DC coefficients using Equation (5.4), and then put the permuted 8×8 quantized blocks back to their original positions;

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

Step-4.1: Perform JPEG's de-quantization process and coefficients redistribution to construct the plain transformed 16×16 block according to key-stream X;

Step-4.2: Do inverse transformation using 16×16 DCT;

Step-5: Repeat *Step-4* until all 16×16 blocks are recovered, add 128 to all pixels to obtain the decrypted image.

Here, we give an example to illustrate how *Algorithm-1-ch5* changes data, compared with the only JPEG compression case. In Figure 5.4, we give two consecutive 16×16 blocks' data $B1$ and $B2$, which are obtained by raster scanning the plain 'Lena' image in 16×16 block unit. The four 8×8 blocks contained in each 16×16 block are denoted as ($B11, B12, B13, B14$) and ($B21, B22, B23, B24$), respectively.

Two 16×16 blocks: $B1$ | $B2$

$B1$

160	160	160	160	160	159	159	159	159	161	159	157	155	152	151	152
160	160	160	160	159	159	158	158	158	160	159	156	155	152	151	152
159	159	159	159	159	158	158	158	158	159	158	155	154	152	151	153
159	159	159	158	158	158	157	157	157	159	157	155	154	152	151	154
158	158	158	158	158	157	157	157	156	158	157	155	154	153	153	155
157	157	157	157	157	156	156	156	156	157	156	154	154	153	154	157
156	156	156	156	156	155	155	155	154	156	155	153	153	153	155	158
155	155	155	155	155	154	154	154	153	155	154	152	153	153	155	158
156	155	155	155	154	154	154	154	155	155	155	154	153	154	156	158
156	156	156	156	156	156	156	157	157	157	157	155	153	153	156	158
156	156	156	156	157	157	158	158	158	159	158	156	154	154	157	159
156	156	156	156	157	158	158	158	158	158	158	156	154	155	159	162
157	157	157	157	158	158	159	159	158	158	158	157	156	157	160	163
157	157	157	157	158	158	158	159	159	159	159	158	159	160	162	163
157	157	157	157	157	158	158	158	159	159	160	161	163	164	165	165
158	158	158	158	159	159	160	160	159	159	160	162	166	168	168	167

$B11$	$B12$
$B13$	$B14$

$B2$

151	155	160	163	165	167	170	172	171	171	168	153	148	144	134	116
152	156	161	164	166	167	170	171	171	171	167	153	148	143	133	115
155	158	162	165	166	168	169	171	170	169	166	152	147	142	132	114
156	159	163	166	167	168	169	170	169	168	164	151	146	141	131	114
157	160	163	166	168	169	169	170	168	167	163	150	145	140	130	113
159	161	164	166	168	169	169	169	167	165	161	149	145	139	129	112
161	163	165	167	169	170	169	169	166	163	159	147	143	138	128	110
163	164	166	168	169	170	169	169	165	162	158	146	142	136	126	109
163	166	167	168	170	170	168	168	162	162	159	147	143	138	127	109
163	166	165	164	165	165	163	163	163	162	160	148	144	139	129	111
165	167	165	163	163	163	161	161	161	161	159	148	144	140	130	112
166	167	166	164	164	163	162	162	161	161	159	148	145	140	130	112
164	166	165	163	164	163	161	162	162	161	160	149	145	141	130	112
164	166	165	163	163	162	160	160	161	161	159	148	145	140	129	111
165	167	165	162	162	162	160	161	159	159	158	147	144	139	128	110
166	167	164	161	161	161	161	162	159	159	158	147	144	139	129	110

$B21$	$B22$
$B23$	$B24$

Figure 5.4: Original data of two 16×16 blocks.

We first use JPEG to compress these two blocks, and the quality factor is set to be 50. The blocks' processing order of JPEG is $B11, B12, B21, B22, B13, B14$,

$B23, B24$. After JPEG's transformation stage and quantization stage, data in these two blocks are shown in Figure 5.5.

*B*1_JPEG

15	0	0	0	0	0	0	0	13	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	15	-1	1	0	0	0	0	0
-1	0	0	0	0	0	0	0	-2	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*B*11	*B*12
*B*13	*B*14

*B*2_JPEG

18	-3	-1	0	0	0	0	0	10	13	-2	1	-1	0	0	0
-1	-1	0	0	0	0	0	0	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	1	0	0	0	0	0	0	8	11	-3	1	-1	0	0	0
1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*B*21	*B*22
*B*23	*B*24

Figure 5.5: Date processed by JPEG.

For *Algorithm-1-ch5*, it realizes encryption during JPEG's transformation stage and after JPEG's quantization stage. In-transformation encryption includes 16×16 DCT transformation, coefficients distribution, and 8×8 blocks' quantization operations, and data in blocks $B1$ and $B2$ processed under these operations are given in Figure 5.6. The processing unit in our proposed scheme is 16×16 size block, thus we obtain all eight 8×8 blocks in the order of $B11, B12, B13, B14, B21, B22, B23, B24$. During the coefficients distribution, $B12$ and $B24$ are the two 8×8 blocks who possess the DC coefficients of their corresponding 16×16 blocks, thus $DCIndex = (2, 8)$.

$B1$_proposed
(In-transformation
encryption)

0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0
1	-1	0	0	0	0	0	0	-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$B11$	$B12$
$B13$	$B14$

$B2$_proposed
(In-transformation
encryption)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	-1	0	0	0	0	0	0	27	17	-14	3	-1	0	0	0
-2	0	0	0	0	0	0	0	2	-1	-2	-1	0	0	0	0
2	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$B21$	$B22$
$B23$	$B24$

Figure 5.6: Date processed by in-transformation encryption.

For the after-quantization encryption, it contains 8×8 blocks permutation and DC coefficients confusion operations. Here, we have eight 8×8 blocks, suppose the permutation vector S' for these blocks is $S' = (6, 3, 7, 8, 5, 1, 2, 4)$, then $DCIndex$ is changed into $DCIndex' = (7, 4)$, which means after block permutation, $B24$ block's DC coefficient occurs before $B12$ block's DC coefficient. Therefore, $B12$ block's DC coefficient will be changed from 29 to $29 \oplus 27 = 6$, $B24$ block's DC coefficient remains unchanged. Data in these two 16×16 blocks after performing the after-quantization encryption techniques are shown in Figure 5.7.

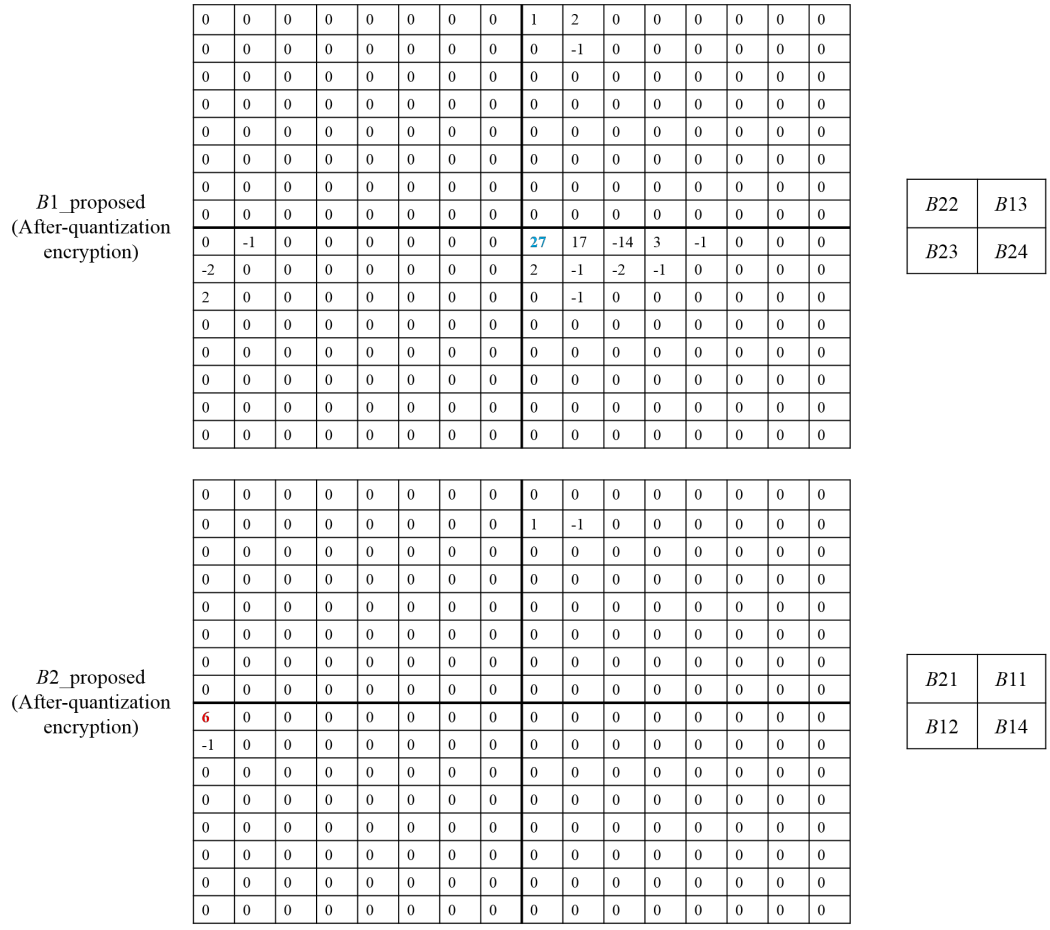


Figure 5.7: Date processed by after-quantization encryption.

5.3 Second encryption scheme

In our first encryption scheme, we do not modify the entropy coding stage of JPEG, which means that to distinguish each 8×8 block, the end-of-block (EOB) identifiers are embedded into the zigzag-scan encoded sequence and this will bring a risk to encryption [80]. In [80], Ji *et al.* pointed out that these position seldom changed identifiers would leak the profile of the plain-image, correlation in 8×8 blocks could not be efficiently removed. Hence they proposed to shuffle the positions of identifiers in the zigzag scan encoded sequence, to make the 8×8 pixel blocks different after encryption. Nevertheless, when shuffling these EOBs, they processed them independently, constraining that each block could only have 63 AC coefficients was not considered appropriate in their cryptosystem, which might lead to the not format-compliant to the JPEG problem, when the encryption key was not available and that only decompression was performed.

In our second encryption method, we add an extra encryption step in the JPEG entropy coding stage adapted based on the first method. The major objective of this added encryption technique is to enhance the correlation removal ability, while keeping the format compliance. Following the idea in [80], we change the position of EOBs by first shuffling the RSV pairs of AC coefficients, then embedding a certain number of EOBs to keep format compliance. The shuffling operation is controlled by key-stream Z. For an image with size being $H \times W$, a total number of $\frac{H \times W}{64}$ EOBs need to be embedded.

5.3.1 Shuffling RSV pairs

In JPEG, after zigzag scanning quantized DCT coefficients, the DC and AC coefficients are separately entropy coded using Huffman coding. For DC coefficients, their differences are encoded. This is because DC coefficients of

neighbouring blocks are highly correlated. While for AC coefficients, they are encoded using variable-length codes (VLC) in the form of RSV pairs. For example, given the following zigzag scan sequence of AC coefficients in two blocks:

$$0, -2, 0, 1, -1, \underbrace{0, \ldots, 0}_{58 \text{ zeros}}, 0, 0, 0, 0, 5, \underbrace{0, \ldots, 0}_{59 \text{ zeros}},$$

The resulting RSV pairs are

$$(1, -2), (1, 1), (0, -1), (0, eob), (3, 5), (0, eob).$$

where $(0, eob)$ is the stopping tag of one 8×8 block, which is essential for ensuring format compliance.

In the second encryption method, we first save all RSV pairs of the non-zero AC coefficients to form a list L^{ac}, the stopping tag $(0, eob)$ is not included in L^{ac}. To show where each 8×8 block ends, we count the number of nonzero AC coefficients in each block, denoted by $\zeta(i, j)$, for $1 \leq i \leq H/8$ and $1 \leq j \leq W/8$. Each element in $\zeta(i, j)$ is mapped to a unique RSV pair in the Huffman table for AC coefficients, using the method proposed in [70]. In the default table for AC coefficients, (Run/Size, Value)=(0, 1) and (0, -1) in category 1 are the shortest codewords, followed by (0, -3), (0, -2), (0, 2), and (0, 3) in category 2, and so forth. To maintain the final encrypted bit-stream size, we represent each of the possible 64 values (since $0 \leq \zeta(i, j) \leq 63$) by a codeword in categories 1, 2, ..., and 6. Specifically, we map (0, -1) to $\zeta(i, j) = 0$, (0, 1) to $\zeta(i, j) = 1$, (0, -3) to $\zeta(i, j) = 2$, (0, -2) to $\zeta(i, j) = 3$, and so on. The mapping results are shown in Figure 5.8. We name these newly generated RSV pairs as L^{acNum}, which is used to distinguish each 8×8 block, and its length is $H \times W/64$. By concatenating L^{acNum} to the end of L^{ac} to form a new list L, we perform a permutation on L using key-stream Z to achieve AC coefficients encryption.

The permutation vector is also produced through the two key-driven cyclical shift, and the permuted RSV pairs' list is denoted as L^p.

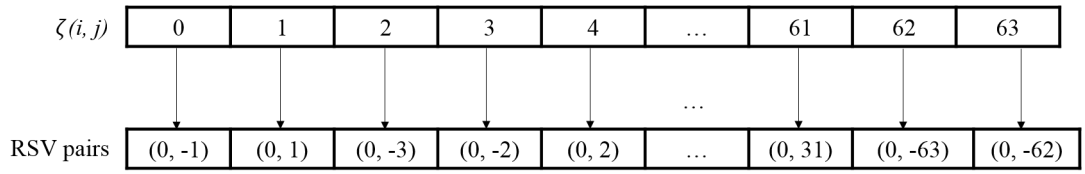


Figure 5.8: Generation of new RSV pairs.

5.3.2 Embedding end-of-block identifiers

Now we have obtained the permuted RSV pairs' list L^p, if the decoder knows key-stream Z, after the JPEG entropy decoding process, it just uses Z to determine the permutation vector to de-permute L^p, and obtains L. By taking the final $H \times W/64$ RSV pairs to distinguish how many nonzero AC coefficients are included in each 8×8 block, the RSV pairs of all nonzero AC coefficients can be easily inserted into their original 8×8 blocks. However, if the secret key is not known, it cannot be distinguished where each 8×8 block will end. As result, the JPEG decompression procedure may not be run successfully, which results in a not format-compliant problem.

To ensure format-compliant when only JPEG de-compression process is performed, we embed $H \times W/64$ EOB identifiers into L^p. These embedded EOBs must satisfy three conditions: 1) the number of AC coefficients in each 8×8 blocks cannot exceed 63; 2) element before the embedded EOB must be nonzero AC coefficient, otherwise some zeros in L^p will be missing when the with-key decryption and decompression procedures are performed; 3) one EOB must be placed at the end of L^p. Our method is to embed the $H \times W/64$ EOBs one by one. For each embedding, we use two arrays: a forward-array and a backward-array, in which the forward-array is to ensure the currently embedded EOB satisfies the above three conditions, while the backward-array is to

make sure that the remaining $(H \times W/64 - 1)$ EOBs can be successfully embedded into the left RSV pairs under the three constraints. Specific realization of our EOB embedding method is given in Pseudo-codes 4.

It is noted that when QF value is high, such as 99 or 100, there exists high possibility that for all quantized 8×8 blocks, their last AC coefficient ($64th$ position after zigzag scan) is non-zero. Under this situation, after we producing a new RSV pair for each 8×8 block that is used to represent $\zeta(i, j)$, the number of AC coefficients in each 8×8 block have already exceeded 63, and there is no room for EOB embedding. To solve this problem, after the quantization process, we first save the last non-zero AC coefficient's position of a 8×8 block, denoted as $LastAC(i, j)$, for $1 \leq i \leq H/8$ and $1 \leq j \leq W/8$. If $\frac{\sum_{i,j} LastAC(i,j)}{(H \times W)/64} > 63$ (DC coefficient's position is contained). We delete the final RSV pair from all 8×8 quantized blocks, so that the number of AC coefficients, including the newly produced RSV pair, in each 8×8 block will not exceed 63. This solution may have a small impact on the final compression efficiency due to the deletion of some information from the plain-image. However, the impact is negligible because the removed information is the high frequency AC coefficients, which only possess minimum energy after DCT transformation.

We take the example in Section 5.3.1 to explain how we obtain the forward-array (denoted as A_f) and the backward-array (denoted as A_b) for each EOB's embedding. After saving all nonzero AC coefficients' RSV pairs and producing new RSV pairs for $\zeta(i, j)$, L is given as follows:

$$\underbrace{(1,-2),(1,1),(0,-1),(3,5)}_{L^{ac}},\underbrace{(0,-2),(0,1)}_{L^{acNum}}.$$

Pseudo-codes 4 End-of-block Identifiers Embedding

Input: Permuted RSV pairs (L^p),
Maximum number of coefficients for 8×8 blocks (T),
EOBs number (Num_{eob})
Output: Encrypted AC sequence with EOBs (S_{ac})

$Num_{avg} \leftarrow \dfrac{length(L^p)}{(H \times W)/64}$;
$T \leftarrow 63$;
$Num_{eob} \leftarrow \dfrac{H \times W}{64}$;
while $Num_{eob} \geq 1$ **do**
  Generate A_f and A_b from L^p;
  $Index_{closest} \leftarrow$ index of element x in A_f satisfying x is the most closest element to Num_{avg};
  $Index_{end} \leftarrow \max(find(A_f \leq T))$;
  $Index_{start} \leftarrow \max(find(A_b > (Num_{eob} - 1) \times T))$;
  if $Index_{closest} > Index_{end}$ **then**
    $Index_{eob} \leftarrow Index_{end}$;
  else
    if $Index_{closest} < Index_{start}$ **then**
      $Index_{eob} \leftarrow Index_{start}$;
    else
      $Index_{eob} \leftarrow Index_{closest}$;
    end if
  end if
  Put the first $A_f(Index_{eob})$ elements of L^p and one EOB identifier into S_{ac};
  $Num_{eob} \leftarrow Num_{eob} - 1$;
  $L^p \leftarrow$ remove the first $A_f(Index_{eob})$ elements from L^p;
end while
return S_{ac}

After performing permutation on L, suppose the permuted RSV list L^p is

$$(0,-1),(0,-2),(1,-2),(1,1),(0,1),(3,5)$$

$$\downarrow$$

$$-1,-2,0,-2,0,1,1,0,0,0,5$$

The newly obtained RSV pairs L^{acNum} is disrupted in this list. By creating a new array A_{num} to count how many coefficients (including zero coefficients) are contained in each RSV pair, A_{num} for the above example is then given as follows:

$$A_{num} = (1,1,2,2,1,4)$$

Generate the forward-array A_f by adding each element in A_{num} with its previous elements starting from A_{num}'s first element, while the backward-array A_b is produced by adding each element in A_{num} with its following elements starting from A_{num}'s last element. Then A_f and A_b will be

$$\begin{aligned} A_f &= (1,2,4,6,7,11), \\ A_b &= (11,10,9,7,5,4), \end{aligned}$$

5.3.3 Encryption and decryption algorithms for the second method

For our second encryption scheme, its encryption and decryption algorithms can be obtained by modifying *Step-4* in *Encryption Algorithm-1-ch5* and *Step-2* in *Decryption Algorithm-1-ch5*, respectively. Other steps remain the same.

Encryption Algorithm-2-ch5

Step-1∼Step-3: Same as *Encryption Algorithm-1-ch5*;

Step-4: Use variable-length codes to represent AC coefficients in the form of RSV pairs, then

Step-4.1: Concatenate all RSV pairs to form a list L_{ac}, in which EOB identifiers are not included;

Step-4.2: Count the number of nonzero AC coefficients in each 8×8 block (ζ), and map these numbers to unique RSV pairs, according to the mapping table, to form another new list L^{acNum} with $H \times W/64$ elements;

Step-4.3: Add L^{acNum} to the end of L_{ac}, denoted as L, apply cyclical shift and key-stream Z to permuting L, and generate a new permuted RSV pairs' list L^p;

Step-4.4: Embed $H \times W/64$ number of EOB identifiers into L^p to obtain the final encrypted AC coefficients with end-of-block labels, denoted as S_{ac};

Step-4.5: Perform JPEG' entropy coding procedure for S_{ac} and DC coefficients, transmit the encrypted bit-stream and encryption keys (256-bit hash value σ, three initial values x'_0, y'_0, and z'_0) securely to decoder for decryption and decompression;

Decryption Algorithm-2-ch5

Step-1: Same as *Decryption Algorithm-1-ch5*;

Step-2: Perform entropy decoding procedure of JPEG, discard all EOB labels, and obtain the permuted RSV pairs' list L^p, then

Step-2.1: Use key-stream Z and cyclical shift to de-permute L^p, and get L;

Step-2.2: Take the final $H \times W/64$ RSV pairs, and map them to the specific number of nonzero AC coefficients (ζ);

Step-2.3: Use ζ to put those remaining RSV pairs in L into their original permuted 8×8 blocks;

Step-2.4: Generate the 8×8 blocks' permutation vector S' using the two

key-driven cyclical shift and key-stream Y;

For decryption without encryption keys, the JPEG decompression steps can be

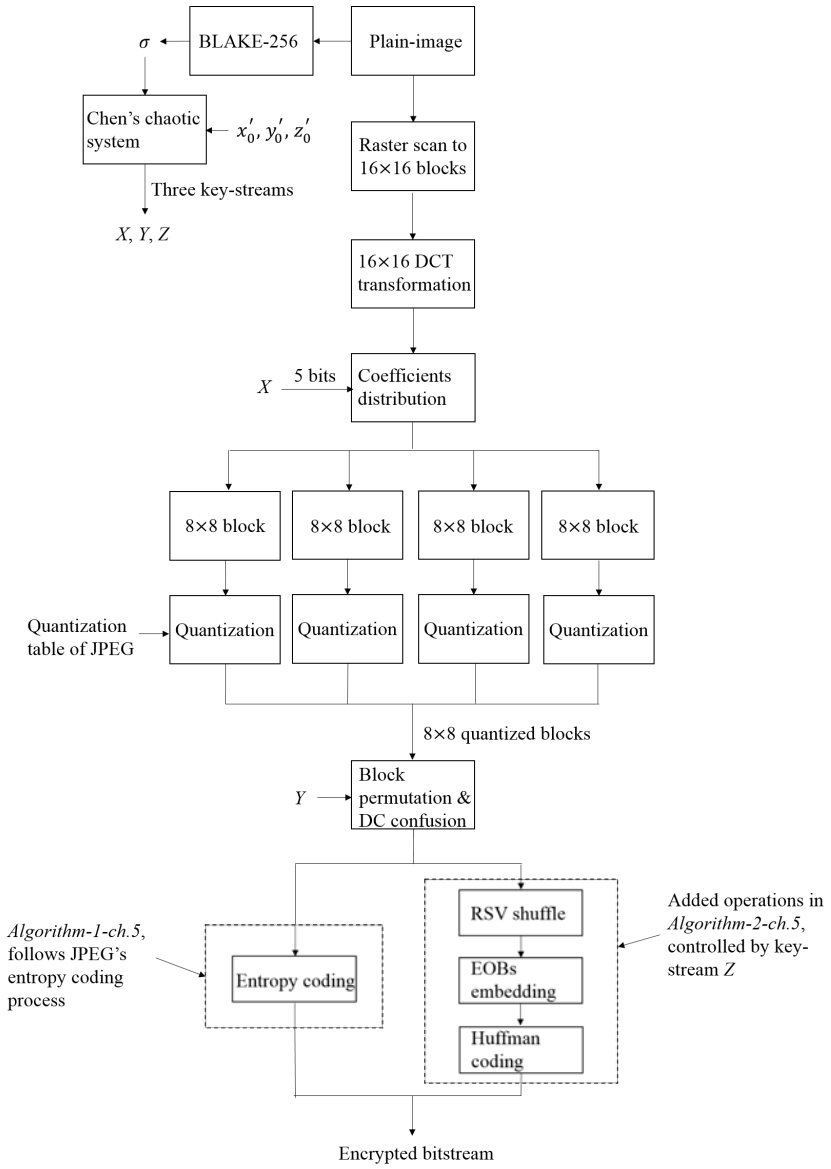


Figure 5.9: Encryption procedures of the two proposed DCT-16-based encryption schemes.

adopted directly onto the encrypted bit-stream because of the added EOB identifiers. The encryption procedures of our first and second encryption schemes are presented in Figure 5.9, and they can be applied for encryption for both grayscale and color images. Flowchart for the decryption process is not given,

since it is just the reverse order of all encryption operations contained in the encryption process flowchart.

5.4 Performance evaluation

In this section, we evaluate the encryption security and compression performance of the proposed two encryption schemes. In our experiment, parameters of Chen's system are: $a = 35$, $b = 3$, and $c = 28$ [122]. The three initial states are: $x'_0 = 10$, $y'_0 = -6$, and $z'_0 = 37$. The same ten grayscale images shown in Figure 4.5 are used for testing. The experimental environment is MATLAB R2014a in 64 bit operating system, 3.50 Ghz, 16 GB RAM, Intel Core i7-4770K. The performance of our proposed encryption schemes is compared with Au Yeung's encryption method [59] and Qian's encryption method [71], since they both realized multimedia data encryption by introducing encryption techniques into the intermediate stages of the underlying compressor.

5.4.1 Encryption security

Multimedia encryption is different from text data encryption, as it requires both perceptual security and cryptographic security [14]. For perceptual security, it refers to the perceptual distortion of cipher-image with respect to the plain-image, which is commonly measured by the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [12]. We encrypt the ten images in Figure 4.5 using our two proposed encryption methods under different QF values, and the average PSNR value and SSIM value of these cipher-images are shown in Figure 5.10. It can be seen that when encryption keys are not known, the PSNR value and SSIM value of cipher-images obtained by *Algorithm-2-ch5* are lower than that of cipher-images encrypted by *Algorithm-1-ch5*, which means a better image distortion ability. The encrypted images and decrypted

images of 'Baboon' and 'Car' under *Algorithm-1-ch5* and *Algorithm-2-ch5* with QF=20 are given in Figure 5.11 as examples.

For the cryptographic security, we evaluate the performance of *Algorithm-1-ch5* and *Algorithm-2-ch5* under four cryptanalysis techniques: brute-force attack, differential attack, key sensitivity analysis, and statistical attack.

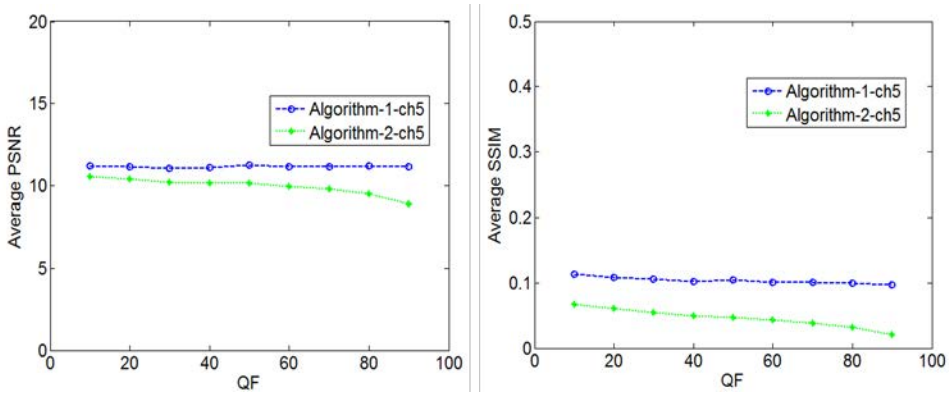


Figure 5.10: Perceptual security results.

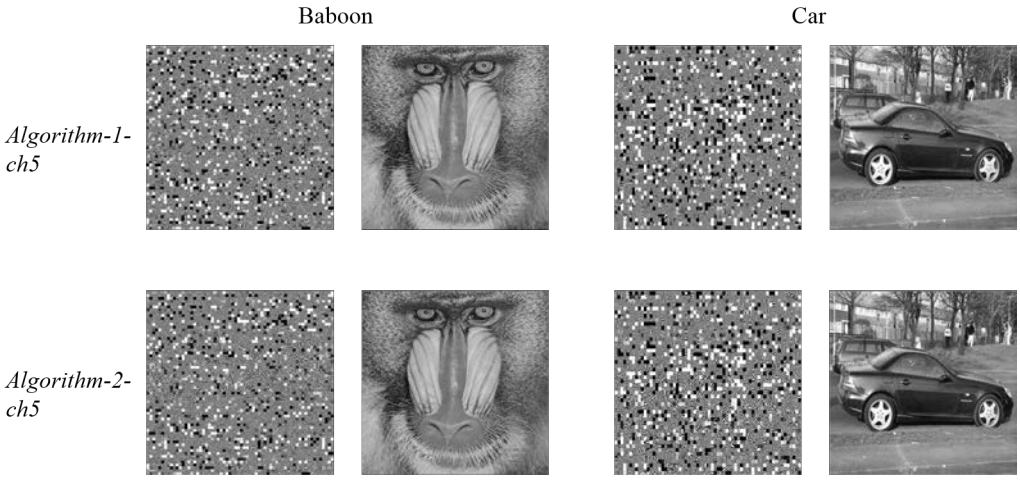


Figure 5.11: Encryption and decryption examples.

- *Brute-force attack.* In both our two encryption schemes, the encryption keys are: 256-bit random hash value σ generated from BLAKE-256, and three initial states of Chen's chaotic system x'_0, y'_0, and z'_0. For the three initial values, if the precision is 10^{14}, the key space size is 10^{42}, thus the total key space size will be $2^{256} \times 10^{42}$, which is impossible for

attackers to guess. Moreover, for each different plain-image, BLAKE-256 will produce a different 256-bit σ, and this also increases the attacking difficulty.

- *Differential attack.* We have computed the NPCR and UACI values for Figure 4.5's ten images with their first pixel increased by 1, and the results for our two encryption algorithms are listed in Table 5.1. Moreover, the NPCR and UACI results of images encrypted by Au Yeung's method [59] and Qian's method [71] are also provided. From Table 5.1, we can observe that both our encryption algorithms have a certain degree of security against differential attack, while for Au Yeung's method and Qian's method, their NPCR and UACI values are almost zero, indicating the low diffusion property. Additionly, it is clear that *Algorithm-2-ch5* generally can obtain higher NPCR and UACI values than *Algorithm-1-ch5*, which means a stronger robustness against differential attack.

Table 5.1: Results of differential attack tests on different images

Image	*Algorithm-1-ch5*		*Algorithm-2-ch5*		Au Yeung's method [59]		Qian's method [71]	
	NPCR%	UACI%	NPCR%	UACI%	NPCR%	UACI%	NPCR%	UACI%
Girl	0.9545	0.2055	0.9675	0.2480	0	0	0	0
Baboon	0.9753	0.2547	0.9538	0.3745	0	0	0	0
Raffia	0.9882	0.1926	0.9839	0.3034	2.4033e-04	3.7399e-06	0	0
Pentagon	0.9834	0.1711	0.9875	0.2566	0	0	0	0
Booth	0.9801	0.2181	0.9744	0.3278	0	0	0	0
Car	0.9744	0.2318	0.9637	0.3295	0	0	0	0
Toy	0.9680	0.2130	0.9731	0.2787	0	0	0	0
Joy	0.9602	0.2133	0.9601	0.2861	0	0	0	0
Sadness	0.9674	0.2144	0.9726	0.2876	0	0	0	0
Surprise	0.9633	0.2119	0.9653	0.2856	7.0190e-04	4.0690e-06	0	0

- *Key sensitivity analysis.* In our proposed two schemes, the encryption keys include two parts: 1) the 256-bit random hash value σ generated from BLAKE-256 and the plain-image; 2) three initial states x'_0, y'_0, and z'_0. In key sensitivity analysis, the plain-image is not changed, thus the 256-bit σ does not change, only the encryption/decryption results come from changes made in these three initial states need to be evaluated. To evaluate the first case of key sensitivity introduced in Section 2.1.2, we slightly change the three initial values $x'_0 = 10$, $y'_0 = -6$, and $z'_0 = 37$ into $x'_0 = 10.00000000000001$, $y'_0 = -6$, and $z'_0 = 37$, and use these two keys to encrypt 'Baboon' image under *Algorithm-1-ch5* and *Algorithm-2-ch5*. The encrypted two images (C1 and C2) by *Algorithm-1-ch5* and *Algorithm-2-ch5* using the two different keys and their difference images are presented in Figure 5.12. For the second case of key sensitivity presented in Section 2.1.2, we use the correct key and slightly changed key to decrypt the cipher Baboon image encrypted by *Algorithm-1-ch5*, and the corresponding results are given in Figure 5.13. As for *Algorithm-2-ch5*, without the correct decryption key, decoder cannot recover the image, since information denoting where each 8×8 block ends is not available.

- *Statistical attack.* We measure the correlation in three directions: horizontal, vertical, and diagonal. In Table 5.2, the correlation coefficients of adjacent pixels in various images encrypted by different encryption schemes are given. From Table 5.2, we can observe that correlations existed in *Algorithm-2-ch5* encrypted images are less than that in plain-images and *Algorithm-1-ch5* encrypted images, which illustrates the second encryption scheme has better decorrelation and robustness against statistical attack, compared with the first encryption scheme.

  For Au Yeung's encryption method, it has better decorrelation perfor-

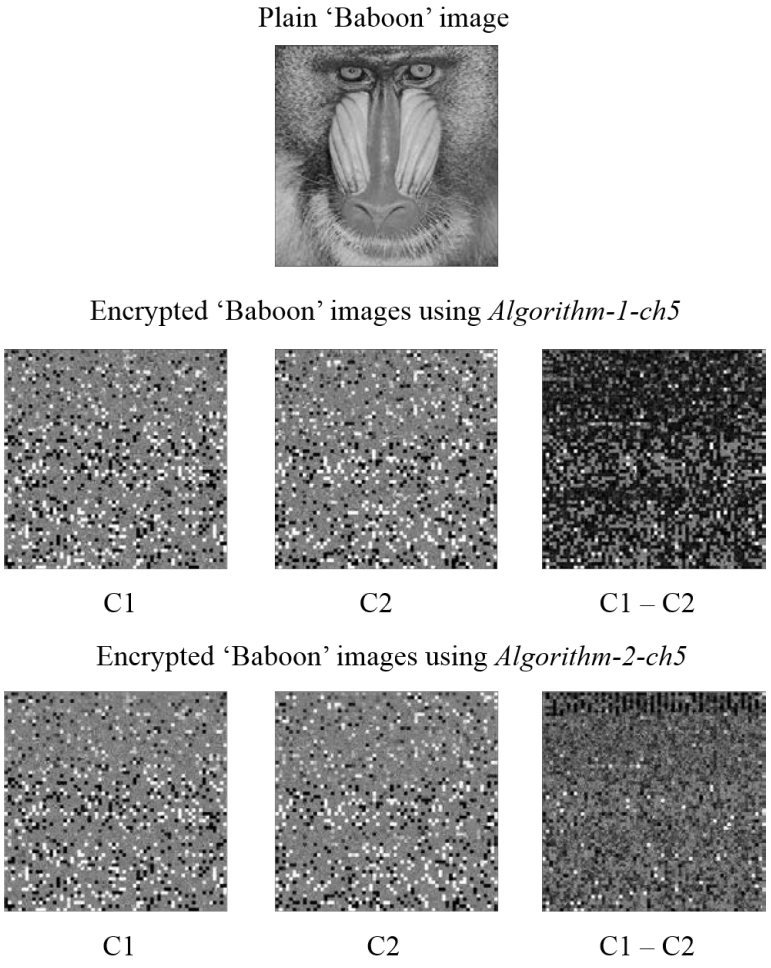


Figure 5.12: Key sensitivity analysis for encryption process.

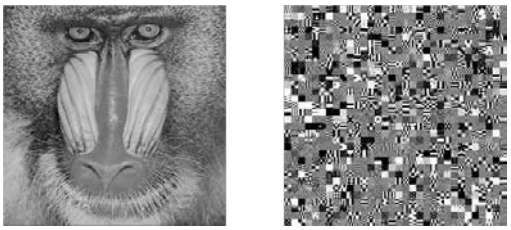


Figure 5.13: Key sensitivity analysis for decryption process: (left) decrypted image via the correct key, (right) decrypted image via the modified key.

mance in vertically and diagonally adjacent pixels than that in horizontally adjacent pixels. This is because for each 8×8 block's transformation, they applied different transform matrix for each column in the second dimension (vertical direction), and used one transform matrix for all eight rows in the first dimension (horizontal direction). Although this operation can decrease pixels correlation, it also compromises the compression efficiency, which will be shown in Section 5.4.3. For images encrypted by Qian's method, the pixels are highly correlated in all the three directions because of their encryption scheme leading to the out of range problem in encrypted pixels values. The correlation charts of plain 'Baboon' image and cipher-images encrypted by various encryption schemes are shown in Figure 5.14.

5.4.2 Encryption efficiency

Apart from the security consideration, efficiency is also an important evaluation criterion for a good image cryptosystem, especially for real-time Internet application. In Table 5.3, we have listed the encryption speed of grayscale images with different sizes by using our proposed two encryption schemes, and they are compared with Au Yeung's method and Qian's method. The JPEG compression-only execution time is also given in the table as a reference. These tested images are chosen from the USC-SIPI image database and RAISE database [124]. The simulation environment is MATLAB R2014a in 64 bit operating system, 3.50 Ghz, 16 GB RAM, Intel Core i7-4770K. From Table 5.3, we can see that the speed of our proposed two encryption schemes is slower than that of Au Yeung's method and Qian's method. This is because they implemented encryption operations only at the JPEG transformation stage or entropy coding stage, resulting in less processing time and lower se-

Table 5.2: Correlation coefficients of adjacent pixels

Image	Original		
	Horizontal	Vertical	Diagonal
Girl	0.9761	0.9735	0.9525
Baboon	0.8597	0.7504	0.7246
Raffia	0.8927	0.7363	0.6890
Pentagon	0.8360	0.8530	0.7776
Booth	0.9238	0.9310	0.9023
Car	0.9332	0.9196	0.8865
Toy	0.9473	0.9412	0.9145
Joy	0.9721	0.9670	0.9618
Sadness	0.9785	0.9788	0.9638
Surprise	0.9693	0.9766	0.9630

Image	*Algorithm-1-ch5*			*Algorithm-2-ch5*		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Girl	0.8213	0.8449	0.7718	0.7352	0.7685	0.6292
Baboon	0.8597	0.7504	0.7246	0.3437	0.3295	0.0582
Raffia	0.8927	0.7363	0.6890	0.2494	0.3986	0.0520
Pentagon	0.8445	0.8319	0.7273	0.4358	0.5335	0.2487
Booth	0.8391	0.8838	0.7122	0.4171	0.4618	0.1170
Car	0.8002	0.8299	0.6644	0.4925	0.5559	0.2280
Toy	0.8728	0.8841	0.7566	0.6599	0.7004	0.5202
Joy	0.8592	0.8295	0.7662	0.6739	0.6670	0.4191
Sadness	0.8831	0.8483	0.7745	0.6452	0.6923	0.3977
Surprise	0.8890	0.8401	0.7535	0.6297	0.6931	0.4066

Image	Au Yeung's method			Qian's method		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Girl	0.8906	0.3707	0.3906	0.9786	0.8787	0.9435
Baboon	0.7600	0.3092	0.2721	0.9830	0.9642	0.9577
Raffia	0.8158	0.3391	0.3072	0.9468	0.9522	0.7219
Pentagon	0.7650	0.3398	0.2965	0.9903	0.9617	0.9845
Booth	0.8093	0.3459	0.3464	0.9397	0.8997	0.8926
Car	0.7932	0.3902	0.2619	0.9355	0.9035	0.7957
Toy	0.8650	0.2932	0.3264	0.9853	0.9746	0.9531
Joy	0.8806	0.4367	0.3792	0.9729	0.9439	0.9463
Sadness	0.8733	0.3723	0.3063	0.9653	0.9171	0.8845
Surprise	0.8901	0.3396	0.3946	0.9608	0.8863	0.8954

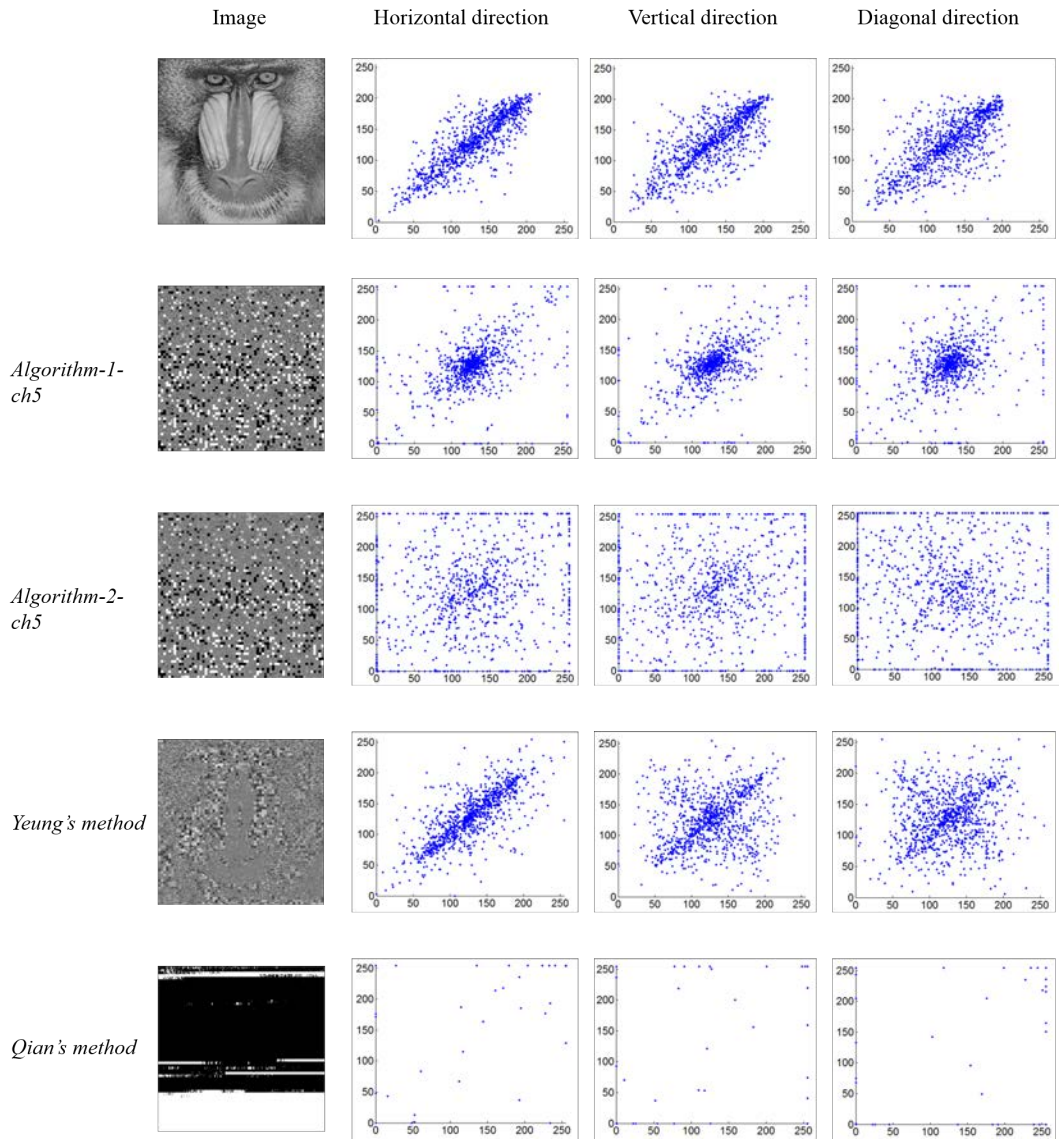


Figure 5.14: Correlation charts of plain 'Baboon' image and encrypted image.

curity. Moreover, *Algorithm-2-ch5*'s encryption speed is a slightly slower than *Algorithm-1-ch5*. This is because in *Algorithm-2-ch5*, in order to enhance the security level and to keep format-compliant, we add the RSV shuffling and EOB labels embedding operations, which lead to longer processing time.

Table 5.3: Efficiency comparison for different encryption schemes

Image size	*Algorithm-1-ch5* (s)	*Algorithm-2-ch5* (s)	Au Yeung's method (s)	Qian's method (s)	JPEG (s)
256×256	1.14	1.40	0.77	0.62	0.40
512×512	4.22	6.44	2.63	1.58	1.21
1024×1024	16.74	28.53	13.34	6.78	5.66
3008×2000	258.71	943.57	276.67	217.09	167.47

5.4.3 Compression performance

Since both our encryption schemes are implemented during the lossy JPEG compression process, the compression performance is influenced by the QF values, and larger QF values result in less compression and larger bitstream sizes. Here, we use the bit per pixel (BPP) value and the PSNR value of decrypted images to evaluate the compression performance of our proposed two encryption schemes. The BPP values and PSNR values of various images with different QF values are shown in Table 5.4 through Table 5.7. These values are compared with the results of JPEG. From the four tables, we can observe that the BPP values of various cipher-images achieved by our two proposed schemes under different QF values are a slightly larger than that of JPEG. Moreover, *Algorithm-2-ch5* has a higher BPP value increment than *Algorithm-1-ch5*, due to some overheads (the embedded $M \times N/64$ EOB identifiers) introduced in the final encrypted bit-stream. The PSNR values achieved by the two methods are also very close to results obtained by JPEG. Based on these results, we believe that *Algorithm-1-ch5* and *Algorithm-2-ch5* are compression friendly to

JPEG.

Table 5.4: Compression performance evaluation when QF = 20

Image	*Algorithm-1-ch5*		*Algorithm-2-ch5*		JPEG	
	BPP	PSNR	BPP	PSNR	BPP	PSNR
Girl	0.3808	32.4928	0.4609	32.4928	0.3647	32.8161
Baboon	0.7477	25.0701	0.8480	25.0701	0.7474	25.2637
Raffia	0.7119	28.6257	0.8149	28.6257	0.7640	28.8030
Pentagon	0.4749	28.9550	0.5573	28.9550	0.4591	29.1710
Booth	0.6592	27.0845	0.7515	27.0845	0.6470	27.3206
Car	0.5897	28.2176	0.6775	28.2176	0.5695	28.5283
Toy	0.4471	30.7578	0.5312	30.7578	0.4175	31.0767
Joy	0.3926	30.6324	0.4740	30.6324	0.3639	30.8930
Sadness	0.3791	30.8992	0.4590	30.8992	0.3558	31.1337
Surprise	0.3905	30.5764	0.4715	30.5764	0.3615	30.7616

Table 5.5: Compression performance evaluation when QF = 40

Image	*Algorithm-1-ch5*		*Algorithm-2-ch5*		JPEG	
	BPP	PSNR	BPP	PSNR	BPP	PSNR
Girl	0.5775	34.6455	0.6629	34.6455	0.5544	35.0026
Baboon	1.2057	27.2773	1.3207	27.2773	1.1935	27.3779
Raffia	1.0956	31.0000	1.2110	31.0000	1.1479	31.2211
Pentagon	0.7961	30.6333	0.8912	30.6333	0.7700	30.8560
Booth	1.0521	29.0908	1.1578	29.0908	1.0276	29.3362
Car	0.9380	30.2771	1.0386	30.2771	0.9028	30.6179
Toy	0.7232	32.8018	0.8145	32.8018	0.6829	33.1936
Joy	0.6425	32.0397	0.7314	32.0397	0.5993	32.3737
Sadness	0.6243	32.2759	0.7118	32.2759	0.5812	32.5732
Surprise	0.6505	32.0081	0.7381	32.0081	0.6021	32.2647

To better illustrate the compression performance of our schemes, we randomly select 30 images from each of the three image databases: JAFFE facial expression database, UCID image database, and USC-SIPI image database, with a total of 90 images for encryption and decryption with QF ranging from

Table 5.6: Compression performance evaluation when QF = 60

Image	*Algorithm-1-ch5*		*Algorithm-2-ch5*		JPEG	
	BPP	PSNR	BPP	PSNR	BPP	PSNR
Girl	0.7733	36.0077	0.8674	36.0077	0.7374	36.3416
Baboon	1.5995	29.0967	1.7229	29.0967	1.5897	29.1476
Raffia	1.4138	32.5587	1.5386	32.5587	1.4678	32.7026
Pentagon	1.1045	31.8165	1.2120	31.8165	1.0607	32.0463
Booth	1.4108	30.6617	1.5273	30.6617	1.3579	30.8723
Car	1.2676	31.8104	1.3767	31.8104	1.2099	32.1191
Toy	0.9647	34.2119	1.0665	34.2119	0.9275	34.6529
Joy	0.9069	33.0364	1.0036	33.0364	0.8454	33.4018
Sadness	0.8950	33.2625	0.9917	33.2625	0.8258	33.5730
Surprise	0.9122	32.9497	1.0100	32.9497	0.8540	33.3407

Table 5.7: Compression performance evaluation when QF = 80

Image	*Algorithm-1-ch5*		*Algorithm-2-ch5*		JPEG	
	BPP	PSNR	BPP	PSNR	BPP	PSNR
Girl	1.1700	38.2137	1.2740	38.2137	1.1329	38.5569
Baboon	2.4119	32.6153	2.5517	32.6153	2.3872	32.5955
Raffia	2.0710	35.2491	2.2047	35.2491	2.1374	35.4462
Pentagon	1.7601	34.0893	1.8844	34.0893	1.6898	34.3139
Booth	2.1275	33.7973	2.2575	33.7973	2.0559	34.0060
Car	1.9099	34.6256	2.0360	34.6256	1.8300	34.8978
Toy	1.4830	36.7231	1.6003	36.7231	1.4212	37.1307
Joy	1.4938	34.9158	1.6098	34.9158	1.4040	35.2248
Sadness	1.4628	35.0491	1.5769	35.0491	1.3782	35.4102
Surprise	1.4984	34.8201	1.6137	34.8201	1.4134	35.2466

10 to 90. Their average BPP-PSNR curves are poltted in Figure 5.15, and compared with JPEG, Au Yeung's method, and Qian's method. From the average BPP-PSNR curves shown in Figure 5.15, it can be seen that Qian's method obtains the best compression performance. This is because that their bitstream encryption method does not change the size of the final encrypted

JPEG bistream. However, it introduces an out of range in pixel values when only decompression is performed on the encrypted bitstream, and the final produced cipher-image will contain large black or white area. When compared with Au Yeung's method, generally speaking, both our encryption schemes can achieve better compression performance, with the *Algorithm-1-ch5*'s BPP-PSNR curve closer to that of JPEG than the *Algorithm-2-ch5*'s curve.

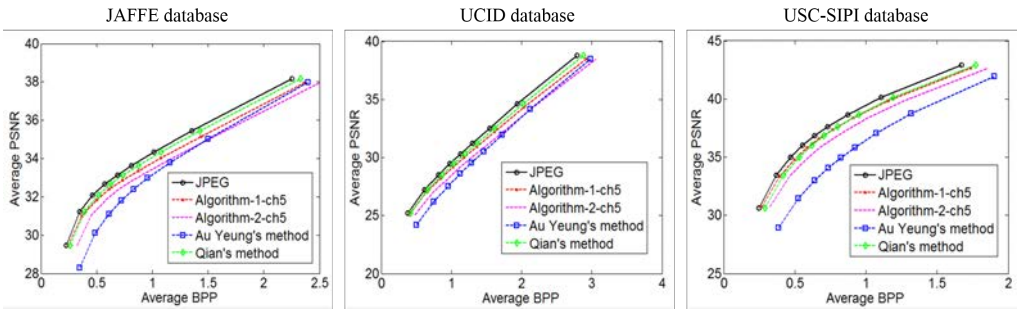


Figure 5.15: Comparion of BPP-PSNR curves for different encryption schemes.

5.5 Summary

In this chapter, we have proposed two image encryption schemes based on 16×16 DCT, which are realized at the intermediate stages of JPEG. They both have their own advantages and disadvantages. For the first encryption scheme, it is more compression friendly to JPEG, but its security level is not so high, hence it may be suitable for applications where compression efficiency is more important than confidentiality. While for the second scheme, it obtains a higher security against the differential attack and statistical attack as compared with the first scheme, but resulting in a slight reduction of compression performance.

In both encryption schemes, we use a adaptive key to control the whole encryption techniques, with the key different for each different plain-image. For the first encryption scheme, we realize encryption in the transformation

stage and after quantization stage of JPEG, which includes the 16×16 DCT transformation, 8×8 block permutation and DC coefficients confusion. For the second scheme, after finishing all encryption operations in the first scheme, we add the entropy coding stage encryption to enhance the security level by shuffling all RSV pairs of nonzero AC coefficients, according to the secret encryption key. Finally, to maintain the format-compliant property, EOB identifiers denoting the side information of each 8×8 block are embedded into these shuffled RSV pairs.

Although the correlation removal ability of the order-16 DCT based cryptosystems is better than the previous order-8 DCT based cryptosystems, the compression efficiency drop is slightly significant. Therefore, in the next step, we will try to combine 8×8 DCT and 16×16 DCT together for image encryption, so as to maintain the compression efficiency of JPEG, meanwhile to obtain good correlation removal ability.

Chapter 6

JPEG image encryption scheme using adaptive block-size

In previous chapters, different cryptosystems have been proposed to realize joint image compression and encryption by using 8×8 size block or 16×16 size block. In this chapter, we use adaptive block-size, 8×8 and 16×16, for JPEG image encryption. Given a plain-image, we first segment it into two different sizes of blocks (8×8 and 16×16), then transform these blocks using order-8/16 DCT and distribute all 16×16 blocks' coefficients into 8×8 blocks, controlled by the encryption key. All transformed 8×8 blocks, either from 8×8 DCT or 16×16 DCT, are shuffled through Fisher-Yates shuffle before quantization. After quantizing all permuted 8×8 blocks, we encrypt quantized AC coefficients by shuffling those RSV pairs with the same run value. These shuffling operations will not destroy the format information of JPEG. For plain-image segmentation, we propose two different methods, in which one is based on the encryption key's information, another is based on plain-image's local intensity information. A comparison between these two different image segmentation methods is provided. The experimental results have shown that both of them can offer a sufficient level of security, and are JPEG compression

friendly.

The reason why I only adopt two block sizes, 8×8 and 16×16 blocks, is because if larger block sizes are applied, such as 32×32 or 64×64 blocks, the out of range problem will be incurred, the same as that described in Section 5.2.1. Moreover, if smaller block sizes are used, such as 4×4 block. The problem of lossing information will be caused, because when we apply order-4 DCT to transform these 4×4 blocks, and then use the JPEG order-8 quantization table to quantize these transformed coefficients, most of them will be zero. When the de-quantization process is applied at the decoder side, many of the coefficients will be zero, and a lot of image information will be lost.

We organize this chapter as follows. In Section 6.1, we explain the implementation details of our encryption operations, and describe the corresponding encryption algorithm and decryption algorithm. Since we use two different methods for image segmentation (key-based, and plain-image-based), this results in two different image encryption schemes. A compression performance evaluation on these two cryptosystems is given in Section 6.2, while their protection powers are analysed in Section 6.3. A summary of this chapter is given in Section 6.4.

6.1 Implementation of encryption operations

In both our encryption schemes, we implement the encryption operations at the JPEG transformation stage and entropy coding stage. In this section, we explain the implementation details of the encryption of these two stages separately. In both schemes, we use a 128-bit predefined secret key as the encryption key, denoted as K_{enc}, and the PRNG adopted here to produce the pseudo-random key-stream is BLAKE2 using Equation (4.1). We denote the BLAKE2 produced key-stream as $K_{enc-pse}$.

6.1.1 Transformation stage encryption

For this stage, the encryption includes three steps: 1) plain-image segmentation; 2) coefficients distribution; and 3) 8×8 transformed blocks permutation.

1) Plain-image segmentation

Our image segmentation method is based on the quadtree decomposition [125]. Given an image, we first raster scan it into non-overlapping 16×16 blocks, then divide this square block into four 8×8 blocks if the segmentation criterion is satisfied. Here, we use two different criteria for the 16×16 block segmentation. The first segmentation criterion is determined by the encryption key $K_{enc-pse}$. We take one bit data from $K_{enc-pse}$, if the bit value is '1', and then divide the 16×16 block into four 8×8 blocks, while '0' means no block splitting. We name this segmentation method as key-based segmentation. The second segmentation criterion is based on the local intensity information of the image, simulation the splitting way in quadtree decomposition. Given a 16×16 image block data, we compute the difference between the maximum pixel value and the minimum pixel value in this block, if the difference is larger than the predefined threshold, denoted as *threshold*, we split this block into four 8×8 blocks, otherwise, no splitting. We name this segmentation method as image-based segmentation.

For quadtree decomposition, different *threshold* values will produce different segmentation results. Figure 6.1 shows three different segmentation results of 'Lena' image under three different *threshold* values. Generally speaking, larger *threshold* results in less number of 8×8 blocks and lower computation complexity. As for threshold value selection, we can use different thresholds for different images segmentation, such as compute the mean value of this block, and set this mean value as the threshold; or use the same threshold value for all images. Here, I adopt the same threshold

value 70 for all grayscale images. The reason for the same threshold value selection is because currently I only realize image encryption by using two size of blocks, 8×8 block and 16×16 block. And in Chapter 5, I have proved that even if the whole images encryption is conducted in 16×16 block unit, the final compression performance does not decrease significantly. Therefore, in the current only two block size based image encryption scheme, the influence of the threshold value on the final compression efficiency is not so obvious. In the future, when I try to realize image encryption by allowing more block sizes, I will consider the problem of how to select the optimal threshold value seriously.

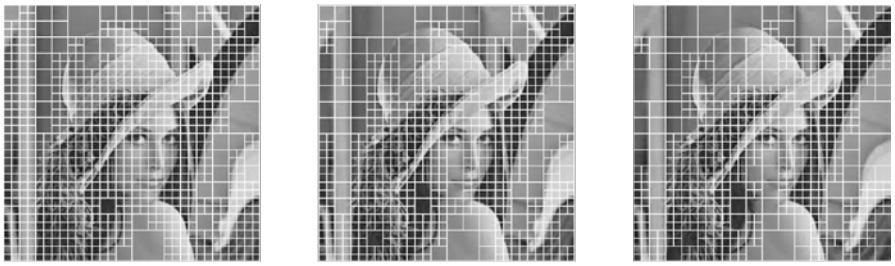


Figure 6.1: Image segmentation results under different threshold values.

To correctly recover the cipher-image, decoder needs to know the plain-image's segmentation method. In the key-based segmentation method, if the encryption key $K_{enc-pse}$ is known, then the encrypted image can be recovered successfully. While in the image-based segmentation method, additional data denoting the segmentation way needs to be saved and transmitted to the decoder for cipher-image recovery. Here, for each 16×16 block, we use 1 bit data to denote whether this block is split or not, where '1' means block splitting, and '0' means no block splitting. Therefore, for an image of size $H \times W$, a total of $\frac{H \times W}{16 \times 16}$ bits are needed to represent the segmentation details. We encrypt these bits using another 128-bit secret key (K_{seg}) to produce a pseudo-random key-stream $K_{seg-pse}$. The final

encrypted segmentation data is denoted as J.

2) Coefficients distribution

In our encryption scheme, we use the JPEG order-8 quantization table. Before quantization, we distribute the 256 coefficients of one 16×16 block into four 8×8 blocks, and the distribution process is the same as Pseudo-codes 3 in Chapter 5.

3) Block permutation

After we finish blocks transformation using order-8/16 DCT and 16×16 blocks' coefficients distribution, we shuffle all 8×8 transformed blocks. The block permutation method we use is also the Fisher-Yates shuffle, described in Chapter 3. Then we quantize these permuted 8×8 blocks using the JPEG original quantization table.

6.1.2 Entropy coding stage encryption

For the JPEG standard, after quantization and roundness, all coefficients in each 8×8 block are integers. The DC coefficients are then encoded using DPCM. The AC coefficients are zigzag scanned into an integer sequence, which contains a significant number of consecutive zeros, which is suitable for run-length coding. In run-length coding, each non-zero AC coefficient is encoded in the form of RSV pair (r, v). After representing all non-zero AC coefficients in RSV pairs for $(\frac{H \times W}{8 \times 8})$ order-8 blocks, we encrypt these RSV pairs by grouping them based on their r value, (r, v) pairs with same r are inserted into the same group. Then we permute each group's (r, v) pairs using the Fisher-Yates shuffle and $K_{enc-pse}$, and this process is named as the same-run shuffle method. This encryption method only changes the v value of each RSV pair, which is still a valid variable length coding (VLC) codeword, thus the final encrypted bit-stream is decodable by the JPEG decoder even without

decryption.

6.1.3 Encryption algorithm and decryption algorithm

Based on the different criteria used for image segmentation, key-based or plain-image-based, there are two different image cryptosystems, in which the key-based segmentation method does not produce extra data to denote segmentation details, while plain-image-based segmentation scheme needs addition data for the decoder so that it knows how the block segmentation is implemented. We first introduce the encryption algorithm and decryption algorithm of the key-based-segmentation cryptosystem, and they are described as follows:

Encryption Algorithm-1-ch6

Step-1: Initialize the BLAKE2 key generator with the 128-bit encryption key K_{enc} to produce the pseudo-random key-streams $K_{enc-pse}$;

Step-2: Raster scan the image into non-overlapping 16×16 blocks, for each block, take one bit data from $K_{enc-pse}$, if its value is '1', segment this block into four 8×8 blocks, otherwise no segmentation;

Step-3: Repeat *Step-2* until all 16×16 blocks' segmentation finishing, transform all 16×16 blocks, and 8×8 blocks using corresponding DCT, and distribute coefficients of 16×16 blocks into 8×8 blocks, controlled by $K_{enc-pse}$;

Step-4: Permute all 8×8 blocks using Fisher-Yates shuffle and $K_{enc-pse}$, and quantize them by JPEG's quantization table;

Step-5: Obtain the RSV pairs (r, v) for all non-zero AC coefficients, shuffle RSV pairs with same r value to realize AC coefficients encryption;

Step-6: Perform JPEG's entropy coding procedure for DC coefficients and AC coefficients, transmit the encrypted bit-stream to decoder for decryption and

decompression.

For decryption without the encryption key K_{enc}, we just follow JPEG's decompression process to obtain the cipher-image, because of the format-compliant property. When key is available, the decryption algorithm is as follows,

Decryption Algorithm-1-ch6

Step-1: Initialize the BLAKE2 key generator with the predefined encryption key K_{enc} to produce the pseudo-random key-streams $K_{enc-pse}$;
Step-2: Perform the entropy decoding procedure of JPEG, and recover the shuffled RSV pairs of AC coefficients through $K_{enc-pse}$;
Step-3: De-quantize all 8×8 blocks by JPEG's quantization table, and put them back to their original positions using Fisher-Yates shuffle and $K_{enc-pse}$;
Step-4: Redistribute coefficients of 8×8 blocks into their original size of blocks, controlled by $K_{enc-pse}$, and perform inverse-transformation process using corresponding order-8/16 DCT;
Step-5: Repeat *Step-4* until we finish all 8×8 blocks' coefficients redistribution work and inverse-transformation work, then we can obtain the decrypted image.

For the image-based-segmentation cryptosystem, some overhead representing image segmentation details needs to be introduced, and another 128-bit secret key K_{seg} are used to encrypt the segmentation data.

Encryption Algorithm-2-ch6

Step-1: Initialize the BLAKE2 key generator with two secret 128-bit keys, K_{enc}

and K_{seg}, to produce two pseudo-random key-streams $K_{enc-pse}$ and $K_{seg-pse}$;

Step-2: Raster scan the image into non-overlapping 16×16 blocks, for each block, do

Step-2.1: Compute the difference between its biggest pixel value and the smallest pixel value;

Step-2.2: Compare the difference with the predefined $threshold$ ($threshold = 70$), determine to segment the 16×16 block into four 8×8 blocks or not;

Step-2.3: Use 1 bit data to represent this 16×16 block's segmentation, '1' means block segmentation, and '0' means no block segmentation;

Step-3: Repeat *Step-2* until all 16×16 blocks' segmentation finishing, encrypt the segmentation data using $K_{seg-pse}$ through bitxor operation, produce the encrypted segmentation data J;

Step-4: Transform all 16×16 blocks, and 8×8 blocks using corresponding DCT, and distribute coefficients of 16×16 blocks into 8×8 blocks, controlled by $K_{enc-pse}$;

Step-5: Permute all 8×8 blocks using Fisher-Yates shuffle and $K_{enc-pse}$, and quantize them by JPEG's quantization table;

Step-6: Obtain the RSV pairs (r, v) for all non-zero AC coefficients, shuffle RSV pairs with same r value to realize AC coefficients encryption;

Step-7: Perform JPEG's entropy coding procedure for DC coefficients and AC coefficients, transmit the encrypted bit-stream, and encrypted segmentation data J securely to decoder for decryption and decompression.

For decryption without the two keys, we just follow JPEG's decompression process to obtain the cipher-image, because of the format-compliant property. When keys are available, the decryption algorithm is as follows,

Decryption Algorithm-2-ch6

Step-1: Initialize the BLAKE2 key generator with the predefined two keys, K_{enc} and K_{seg}, to produce two pseudo-random key-streams $K_{enc-pse}$ and $K_{seg-pse}$;
Step-2∼*Step-3:* Same as *Decryption Algorithm-1-ch6*;
Step-4: Decrypt the encrypted segmentation data J through bitxor operation with $K_{seg-pse}$, and redistribute coefficients of 8×8 blocks into their original size of blocks, and perform inverse-transformation process using corresponding order-8/16 DCT;
Step-5: Repeat *Step-4* until we finish all 8×8 blocks' coefficients redistribution work and inverse-transformation work, then we can obtain the decrypted image.

The encryption procedures of *Algorithm-1-ch6* and *Algorithm-2-ch6* are presented in Figure 6.2. The decryption procedures are not given, since they are just the reverse order of all encryption operations contained in the encryption process flowchart.

6.2 Experimental results

In this section, the perceptual security and the compression performance of our proposed two cryptosystems are evaluated experimentally. Four different sizes of images are tested: 256×256, 384×512 or 512×384, 512×512, and 1024×1024. Because there exists same number of 8×8 blocks in 384×512 size images and 512×384 images, we classify these two sizes of image into the same category. We randomly select these four sizes of images from three image databases: JAFFE facial expression database, UCID image database, and USC-SIPI image database, with each type has 30 same size images. Here, we choose ten grayscale images, shown in Figure 6.3, as experiment samples. We

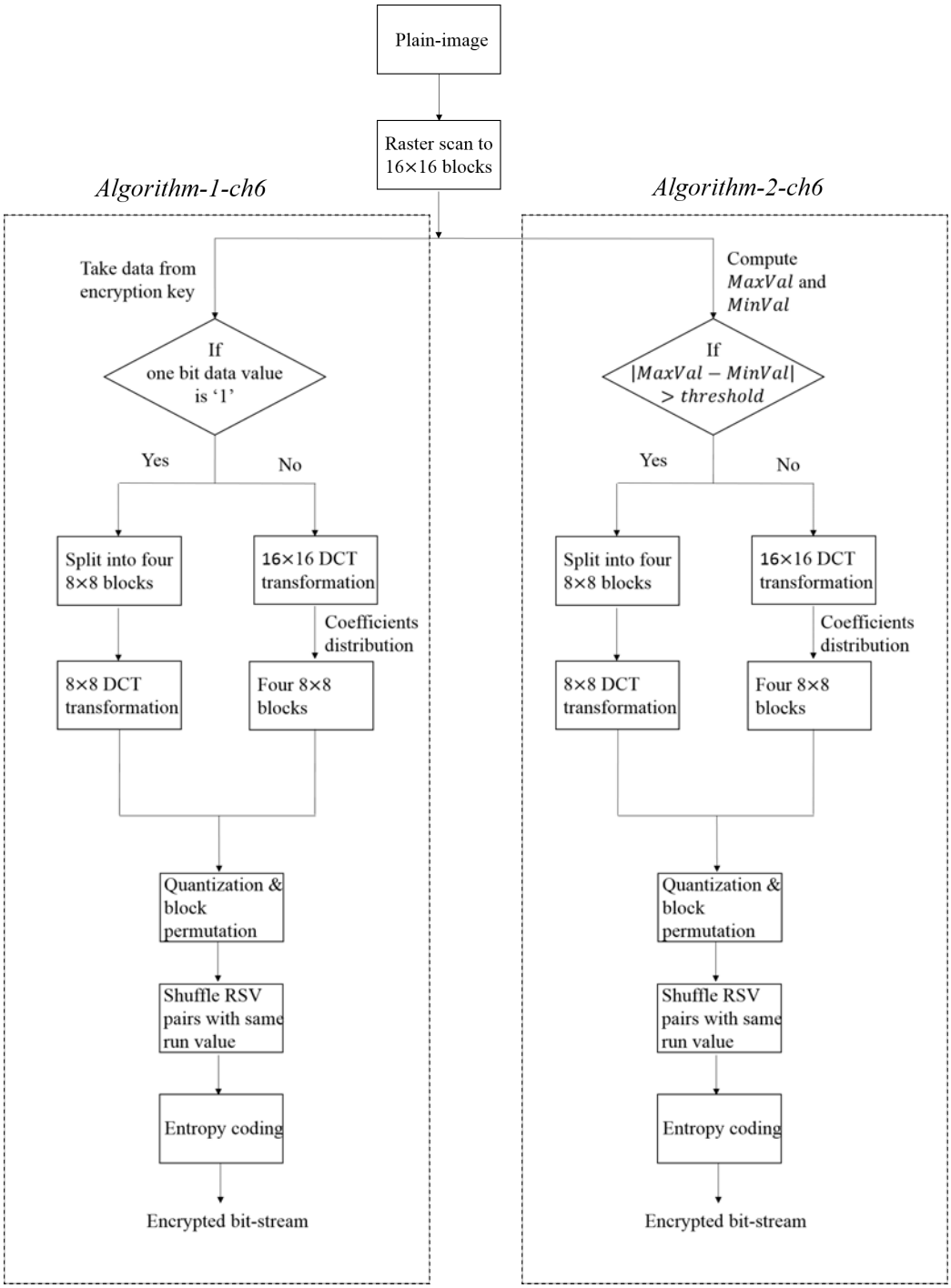


Figure 6.2: Encryption procedures of the two proposed adaptive block size based encryption schemes.

encrypt these images using our proposed two image encryption schemes with QF value ranging from 15 to 95, and the average PSNR value and SSIM value of these cipher-images are shown in Figure 6.4. From Figure 6.4, we can observe that when secret keys are not available to decoders, the PSNR value and SSIM value of cipher-images obtained by *Algorithm-1-ch6* and *Algorithm-2-ch6* are both very low, and these values are decreasing with QF values increasing. This is because the number of non-zero quantized AC coefficients increases with increasing QF values, and our proposed same-run shuffle method operates on more RSV pairs, which makes the final cipher-image more randomised.

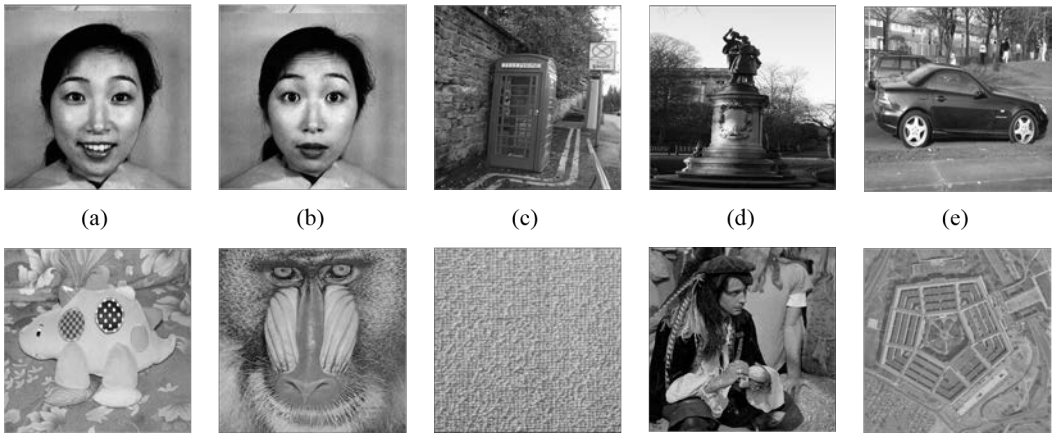


Figure 6.3: Ten test images. (a) Joy 256×256. (b) Surprise 256×256. (c) Booth 384×512. (d) Statue 384×512. (e) Car 512×384. (f) Toy 512×384. (g) Baboon 512×512. (h) Raffia 512×512. (i) Man 1024×1024. (i) Pentagon 1024×1024.

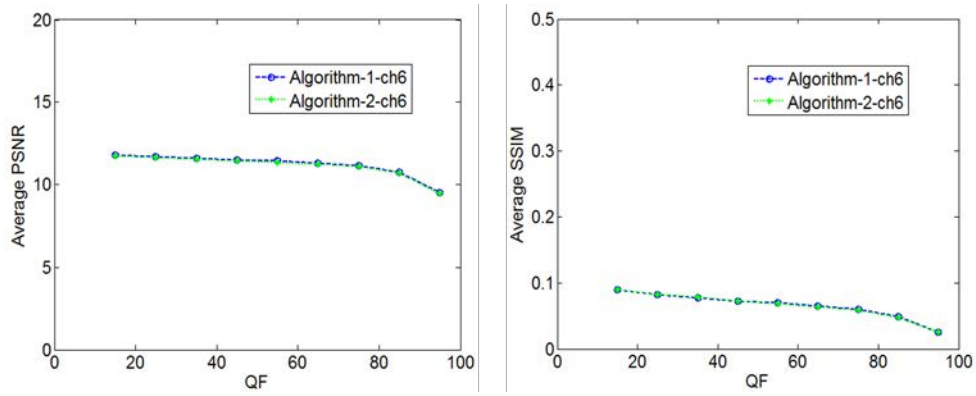


Figure 6.4: Perceptual security results.

For compression, it is influenced by the quality factor (QF). A higher QF value results in less compression and larger bitstream size. Here, we also use the BPP value and PSNR value of cipher-images as evaluation criteria, as mentioned in Chapter 4. The BPP and PSNR values for the ten tested images with different QF values (QF = 15, QF = 45, QF = 75, and QF = 95) are presented in Tables 6.1 through 6.4, and they are compared with the results of JPEG. From the four tables, we can observe that the BPP values of various cipher-images achieved by our proposed two methods under different QF values are slightly larger than results of JPEG, while their PSNR values are nearly the same. This confirms that both our encryption schemes are compression friendly to JPEG. To better show the good compression performance of our two algorithms, we encrypt all 120 different sizes plain-images using *Algorithm-1-ch6* and *Algorithm-2-ch6*. Their average PSNR values under with key situation and without key situation are given in Figure 6.5. They are then compared with JPEG and *Algorithm-1-ch4* proposed in Chapter 4. From Figure 6.5, we can observe that both the proposed schemes can provide good compression performance. For *Algorithm-2-ch6* (image-based-segmentation cryptosystem), although it produces extra data to denote the image segmentation details, the final encrypted bit-stream size under this algorithm does not increase significantly. This is because in *Algorithm-2-ch6*, the segmentation method takes the plain-image's correlation into consideration, as pixels in the segmented 8×8 blocks and 16×16 blocks are highly correlated, hence the compression efficiency is not compromised.

In Table 6.5, we have shown the encryption efficiency of various encryption schemes for encrypting grayscale images with different sizes. The tested images are selected from the USC-SIPI image database. The simulation environment is MATLAB R2014a in 64 bit operating system, 3.50 Ghz, 16 GB RAM, Intel Core i7-4770K. From this table, we can observe that the compu-

Table 6.1: Comparison of compression performance when QF = 15

Image	*Algorithm-1-ch6*		*Algorithm-2-ch6*		JPEG	
	BPP	PSNR	BPP	PSNR	BPP	PSNR
Joy	0.3258	30.0872	0.3288	30.2877	0.3025	30.2147
Surprise	0.3265	29.9992	0.3271	30.2082	0.3008	30.1086
Booth	0.5427	26.4208	0.5485	26.5175	0.5263	26.5154
Statue	0.4609	26.3244	0.4584	26.4305	0.4120	26.4033
Car	0.4881	27.5687	0.4965	27.6874	0.4656	27.6727
Toy	0.3655	30.0438	0.3679	30.2111	0.3372	30.1898
Baboon	0.6112	24.3895	0.6302	24.5060	0.6086	24.5034
Raffia	0.6183	27.7118	0.6338	27.7331	0.6265	27.7331
Man	0.3732	30.3978	0.3776	30.5160	0.3468	30.4750
Pentagon	0.3823	28.3610	0.3879	28.4664	0.3700	28.4511

Table 6.2: Comparison of compression performance when QF = 45

Image	*Algorithm-1-ch6*		*Algorithm-2-ch6*		JPEG	
	BPP	PSNR	BPP	PSNR	BPP	PSNR
Joy	0.6986	32.5018	0.6955	32.6486	0.6601	32.6553
Surprise	0.7030	32.4080	0.6965	32.5611	0.6638	32.5536
Booth	1.1379	29.6506	1.1404	29.7351	1.1110	29.7354
Statue	0.9473	29.7002	0.9343	29.8040	0.8700	29.8074
Car	1.0137	30.8700	1.0158	30.9967	0.9828	31.0083
Toy	0.7833	33.4109	0.7761	33.5592	0.7454	33.5733
Baboon	1.3030	27.7705	1.3151	27.8222	1.2907	27.8227
Raffia	1.2044	31.5676	1.2313	31.6267	1.2232	31.6267
Man	0.7579	33.7075	0.7586	33.8239	0.7242	33.8376
Pentagon	0.8630	31.0750	0.8619	31.1499	0.8437	31.1666

tational requirements of the four proposed schemes are less than the JPEG compression then AES encryption scheme, although it can achieve the best image scrambling effect. For a practical application which emphasizes on the computational requirement, I believe that the proposed *Algorithm-1-ch3* and *Algorithm-1-ch4* will be more preferable because of their simple implementa-

Table 6.3: Comparison of compression performance when QF = 75

Image	*Algorithm-1-ch6*		*Algorithm-2-ch6*		JPEG	
	BPP	PSNR	BPP	PSNR	BPP	PSNR
Joy	1.2332	34.3768	1.2289	34.5522	1.1792	34.5439
Surprise	1.2502	34.3771	1.2383	34.5701	1.1951	34.5577
Booth	1.8406	32.7727	1.8337	32.8329	1.7955	32.8352
Statue	1.5045	33.4224	1.4839	33.5640	1.4043	33.5829
Car	1.6490	33.7761	1.6440	33.8993	1.5995	33.9184
Toy	1.2870	36.0438	1.2726	36.2305	1.2371	36.2826
Baboon	2.1085	31.3484	2.1144	31.3366	2.0833	31.3404
Raffia	1.8643	34.3987	1.8889	34.4396	1.8796	34.4396
Man	1.2403	35.9816	1.2386	36.0735	1.1936	36.0977
Pentagon	1.4805	33.3794	1.4736	33.4571	1.4502	33.4775

Table 6.4: Comparison of compression performance when QF = 95

Image	*Algorithm-1-ch6*		*Algorithm-2-ch6*		JPEG	
	BPP	PSNR	BPP	PSNR	BPP	PSNR
Joy	3.5260	42.5084	3.4948	42.5714	3.4410	42.5985
Surprise	3.5654	42.4693	3.5139	42.4992	3.4645	42.5338
Booth	4.2048	42.9969	4.1782	43.0125	4.1244	43.0390
Statue	3.4021	43.9043	3.3574	44.0095	3.2412	44.0892
Car	3.9696	42.7544	3.9465	42.7998	3.8845	42.7953
Toy	3.2236	43.7028	3.1972	43.7429	3.1359	43.7581
Baboon	4.7370	42.4168	4.7529	42.4164	4.7117	42.4002
Raffia	4.2406	43.0976	4.2664	43.1701	4.2550	43.1701
Man	3.3722	42.5255	3.3614	42.8428	3.2963	42.8699
Pentagon	3.8934	42.3819	3.8810	42.3836	3.8435	42.3904

tion. For the two adaptive block size based schemes, *Algorithm-1-ch6* and *Algorithm-2-ch6*, their computation cost is dependent on the quality factor value, larger QF value means higher computation cost, since the same-run shuffle method will be operated on more AC coefficients.

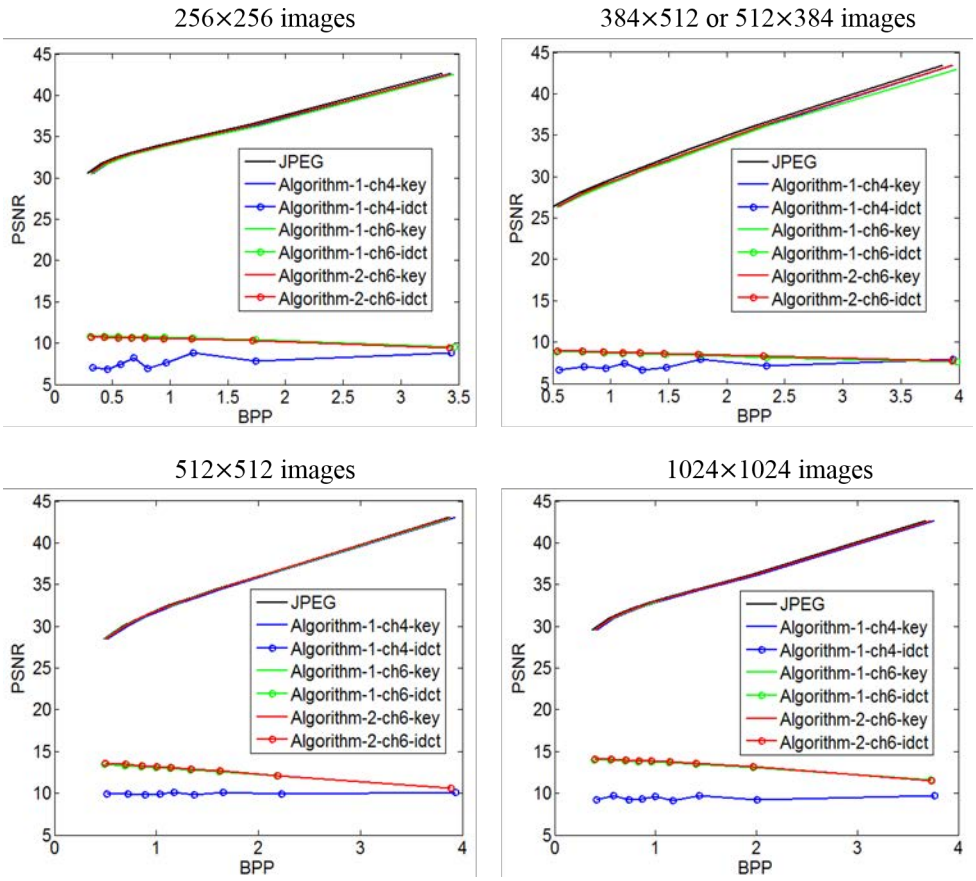


Figure 6.5: Compression performance of the proposed two schemes.

Table 6.5: Efficiency comparison for different encryption schemes

Image size	*Algorithm-1-ch6* (s)	*Algorithm-2-ch6* (s)	*Algorithm-1-ch3* (s)	*Algorithm-1-ch4* (s)
256×256	1.19	1.25	0.65	0.98
512×512	4.05	4.22	2.29	2.64
1024×1024	13.88	22.16	7.94	8.33
Image size	*Algorithm-1-ch5* (s)	*Algorithm-2-ch5* (s)	JPEG compression then AES (s)	JPEG (s)
256×256	1.14	1.40	23.72	0.63
512×512	4.22	6.44	97.57	1.44
1024×1024	16.74	28.53	500.73	3.61

6.3 Security analysis

In this section, we analyse the encryption power of our proposed two cryptosystems by evaluating their robustness against four cryptanalysis techniques: brute-force attack, key sensitivity analysis, statistical attack, and sketch attack.

6.3.1 Brute-force attack

In the key-based-segmentation algorithm *Algorithm-1-ch6*, only one 128-bit key K_{enc} is used for encryption, thus obtain a 2^{128} key space, which is infeasible for attackers to guess. For the image-based-segmentation algorithm *Algorithm-2-ch6*, its key space is 2^{256}, because two 128-bit keys (K_{enc} and K_{seg}) are adopted for image data encryption and segmentation data encryption.

6.3.2 Key sensitivity analysis

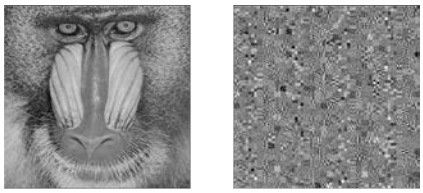
In the proposed two cryptosystems, we use BLAKE2 to produce the corresponding pseudo-random key-streams. Since the output of the BLAKE2 hash function is very sensitive to its input, any minor change occurred in the input results in dramatically different key-stream generated. To test the key sensitivity of our schemes, we slightly modify the encryption key K_{enc} by xor'ing its last bit with 1. The correlation coefficients between different images encrypted by the original key and the slightly changed key are listed in Table 6.6, with QF set to be 75. Here, this QF value is taken as an example. The low correlation coefficients obtained by *Algorithm-1-ch6* and *Algorithm-2-ch6* prove that both the proposed schemes have good encryption key sensitive property. To test the final decrypted cipher-image's sensitivity with regards to decryption key's change, we take 'Baboon' image as an example. We first encrypt the plain 'Baboon' image using *Algorithm-1-ch6* and *Algorithm-2-ch6*, then de-

crypt the cipher-image by the correct decryption key and the slightly changed decryption key. Results are shown in Figure 6.6, from which we can observe that even when there is a small variation in the decryption key, the correct decryption cannot be realized in both the two cryptosystems.

Table 6.6: Correlation coefficient for image encrypted with original key and slightly different key

Algorithm / Image	*Algorithm-1-ch6*	*Algorithm-2-ch6*
Joy	-0.0041	0.0272
Surprise	-0.0015	0.0001
Booth	-0.0001	0.0062
Statue	-0.0015	0.0060
Car	0.0086	0.0097
Toy	-0.0327	0.0101
Baboon	-0.0060	-0.0029
Raffia	0.0119	-0.0021
Man	-0.0016	-0.0082
Pentagon	-0.0051	0.0014

Algorithm-1-ch6



Algorithm-2-ch6

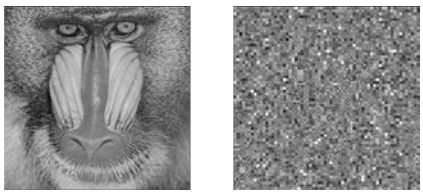


Figure 6.6: Key sensitivity analysis for decryption process: left image decrypted with right decryption key, right image decrypted with slightly changed decryption key.

6.3.3 Statistical attack

In Chapter 3, we have introduced that histograms and correlation diagrams are two common ways to show the relationship between the plain-image and cipher-image. Here, we compute the adjacent pixel correlation coefficients of various plain-images and cipher-images under three different QF values (QF = 55, 75, and 95), and the results are listed in Table 6.7 through Table 6.9. From the three tables, we can observe that both encryption schemes have better security against the statistical attack, as compared with *Algorithm-1-ch4*. Their image decorrelation ability are increasing along with the increased QF values, which is because the same-run shuffle method in *Algorithm-1-ch6* and *Algorithm-2-ch6* is operated on more non-zero AC coefficients. In Figure 6.7 and Figure 6.8, the correlation charts of encrypted 'Baboon' images under different QF values are given. The correlation charts are plotted by randomly selecting 1000 pairs of two horizontal adjacent pixels, two vertical adjacent pixels, and two diagonal adjacent pixels from the encrypted 'Baboon' image. We can see that the linear property, which denotes the correlation degree between pixels, shown in cipher-images encrypted by *Algorithm-1-ch6* and *Algorithm-2-ch6* is decreasing with increasing QF values.

6.3.4 Sketch attack

In this section, we analyse the proposed two cryptosystems' robustness against the sketch attack. Three sketch attack methods in [113] are tested: nonzero coefficients count (NCC), energy of AC coefficients (EAC), and position of last nonzero coefficients (PLZ), as follows:

1) NCC: The number of non-zero AC coefficients in each 8×8 quantized block

Table 6.7: Correlation coefficients of adjacent pixels with QF = 55

Image	JPEG			*Algorithm-1-ch6*		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Joy	0.9916	0.9959	0.9863	0.8323	0.8043	0.6795
Surprise	0.9908	0.9947	0.9846	0.7567	0.5753	0.4122
Booth	0.7285	0.6638	0.5476	0.5200	0.9057	0.4981
Statue	0.9959	0.9985	0.9955	0.7873	0.9072	0.7172
Car	0.7795	0.8328	0.7257	0.6804	0.6997	0.4652
Toy	0.9460	0.9680	0.9232	0.7955	0.9067	0.7235
Baboon	0.4911	0.6261	0.4169	0.3288	0.3528	0.0950
Raffia	0.8537	0.8616	0.7247	0.5558	0.3644	0.1353
Man	0.9930	0.9977	0.9926	0.7742	0.8644	0.6912
Pentagon	0.8221	0.9111	0.7872	0.5067	0.6919	0.3315
Average	**0.8592**	**0.8850**	**0.8084**	**0.6538**	**0.7072**	**0.4749**
Image	*Algorithm-2-ch6*			*Algorithm-1-ch4*		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Joy	0.7863	0.9700	0.7555	0.8951	0.9945	0.8994
Surprise	0.5921	0.9048	0.5248	0.9241	0.9933	0.9294
Booth	0.6322	0.7701	0.6381	0.8647	0.9929	0.8617
Statue	0.7305	0.8846	0.7190	0.8347	0.9766	0.8315
Car	0.6934	0.7680	0.5958	0.8664	0.9709	0.8508
Toy	0.7670	0.8553	0.6854	0.8155	0.9881	0.8112
Baboon	0.3651	0.3007	0.1170	0.8687	0.9329	0.8382
Raffia	0.4418	0.4433	0.0977	0.8785	0.9652	0.8580
Man	0.7801	0.8272	0.6884	0.8941	0.9977	0.8929
Pentagon	0.6162	0.7105	0.4754	0.8188	0.9564	0.7968
Average	**0.6405**	**0.7435**	**0.5297**	**0.8661**	**0.9769**	**0.8570**

Table 6.8: Correlation coefficients of adjacent pixels with QF = 75

Image	JPEG			*Algorithm-1-ch6*		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Joy	0.9873	0.9946	0.9825	0.7528	0.8143	0.6264
Surprise	0.9904	0.9939	0.9832	0.6559	0.7101	0.5517
Booth	0.7119	0.6534	0.5377	0.4771	0.7489	0.4011
Statue	0.9953	0.9976	0.9953	0.6890	0.8039	0.7165
Car	0.7737	0.8270	0.7315	0.5461	0.5876	0.3332
Toy	0.9435	0.9685	0.9191	0.6438	0.6681	0.4917
Baboon	0.5232	0.6193	0.4275	0.1354	0.2741	-0.0783
Raffia	0.8544	0.8389	0.6975	0.4558	0.4583	0.1329
Man	0.9933	0.9970	0.9919	0.6870	0.8546	0.6952
Pentagon	0.8098	0.8872	0.7688	0.4082	0.6269	0.2609
Average	**0.8583**	**0.8777**	**0.8035**	**0.5451**	**0.6547**	**0.4131**
Image	*Algorithm-2-ch6*			*Algorithm-1-ch4*		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Joy	0.8110	0.5821	0.4631	0.8835	0.9896	0.8912
Surprise	0.5439	0.7879	0.4487	0.8737	0.9866	0.8720
Booth	0.6028	0.6719	0.4717	0.8346	0.9534	0.8176
Statue	0.7697	0.8172	0.7218	0.7964	0.9233	0.7612
Car	0.5661	0.4203	0.3681	0.8263	0.9574	0.8226
Toy	0.7062	0.6762	0.6492	0.8076	0.9884	0.8078
Baboon	0.1883	0.1724	0.0014	0.8556	0.9388	0.8203
Raffia	0.3712	0.1672	-0.2443	0.8767	0.9791	0.8629
Man	0.7625	0.8821	0.7285	0.8805	0.9931	0.8779
Pentagon	0.4905	0.4774	0.2405	0.8158	0.9248	0.7832
Average	**0.5812**	**0.5655**	**0.3849**	**0.8451**	**0.9632**	**0.8317**

Table 6.9: Correlation coefficients of adjacent pixels with QF = 95

Image	JPEG			*Algorithm-1-ch6*		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Joy	0.9837	0.9901	0.9798	0.2794	0.2043	0.1354
Surprise	0.9854	0.9860	0.9770	0.3183	0.3091	0.2977
Booth	0.7369	0.6617	0.5552	0.1724	0.2484	0.2570
Statue	0.9967	0.9982	0.9960	0.4310	0.4412	0.4658
Car	0.7978	0.8329	0.7342	0.0849	0.1315	0.1273
Toy	0.9450	0.9669	0.9210	0.3983	0.3753	0.2757
Baboon	0.5415	0.6423	0.4395	-0.2062	-0.1949	0.0049
Raffia	0.8554	0.8415	0.7103	-0.2300	0.0375	-0.2063
Man	0.9903	0.9935	0.9881	0.3713	0.3502	0.2333
Pentagon	0.7880	0.8331	0.7477	-0.1241	-0.0205	0.0370
Average	**0.8621**	**0.8746**	**0.8049**	**0.1495**	**0.1882**	**0.1628**
Image	*Algorithm-2-ch6*			*Algorithm-1-ch4*		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Joy	0.2351	0.0745	0.0259	0.8799	0.9844	0.8890
Surprise	0.1567	0.2725	0.1795	0.8753	0.9875	0.8775
Booth	-0.0179	0.3097	0.0785	0.8344	0.9504	0.8068
Statue	0.6002	0.6972	0.5675	0.8222	0.9273	0.7967
Car	0.1865	0.0580	-0.0005	0.8332	0.9589	0.8278
Toy	0.3328	0.4489	0.2393	0.8403	0.9876	0.8412
Baboon	-0.1613	-0.3398	0.0784	0.8612	0.9380	0.8219
Raffia	0.0089	-0.0736	-0.2709	0.8729	0.9770	0.8568
Man	0.3154	0.4787	0.3171	0.8861	0.9928	0.8825
Pentagon	-0.0614	-0.1693	0.1015	0.8089	0.9221	0.7753
Average	**0.1595**	**0.1757**	**0.1316**	**0.8514**	**0.9626**	**0.8376**

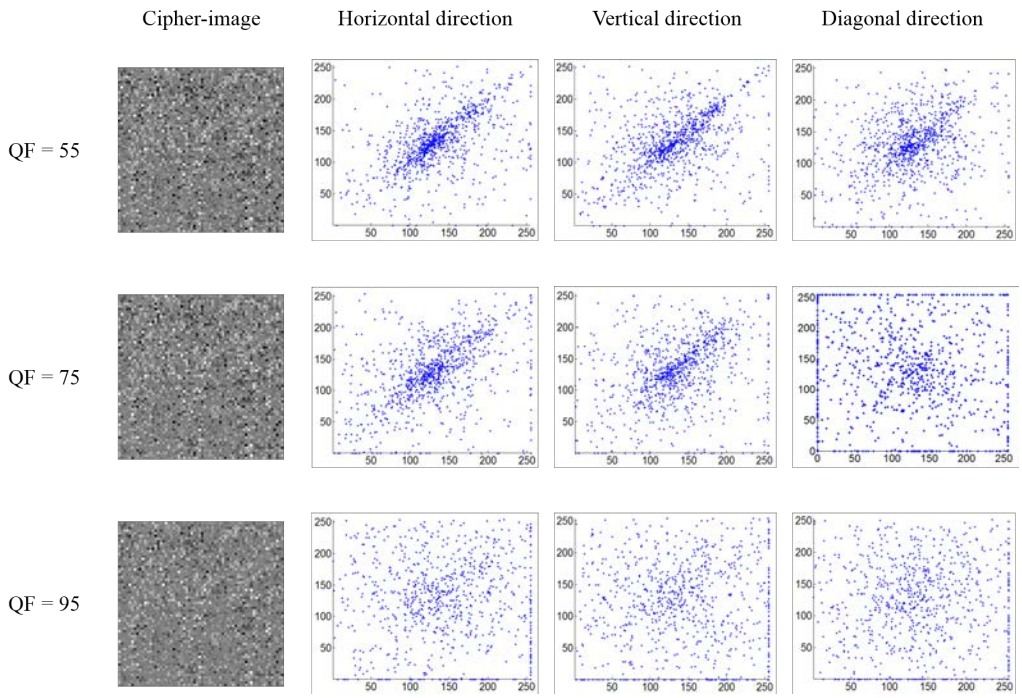


Figure 6.7: Correlation charts of encrypted 'Baboon' image under *Algorithm-1-ch6*.

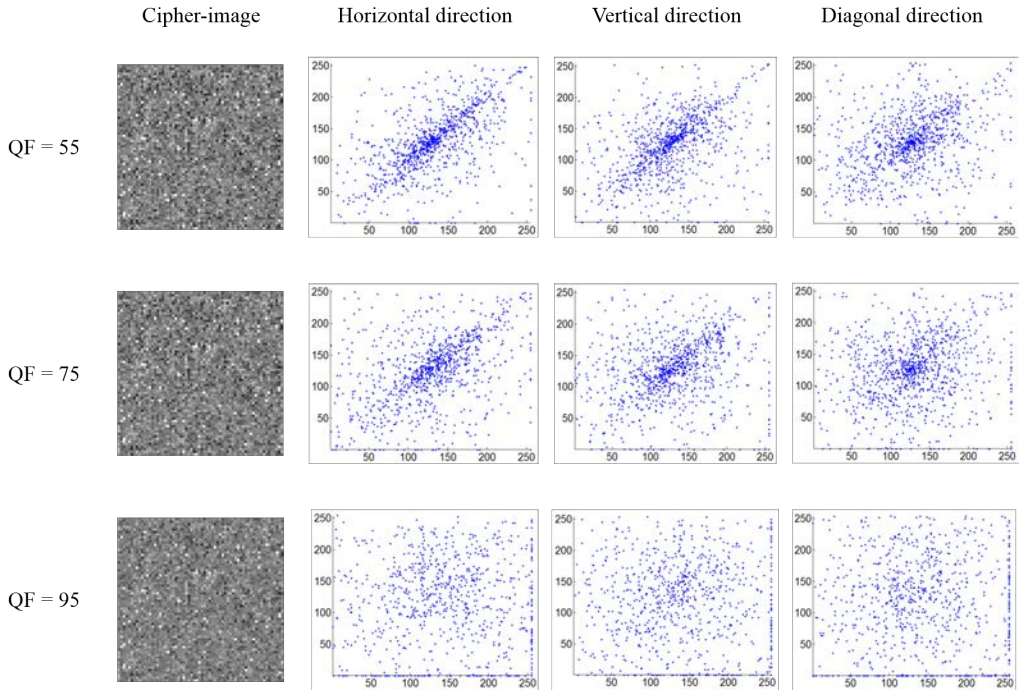


Figure 6.8: Correlation charts of encrypted 'Baboon' image under *Algorithm-2-ch6*.

is exploited to generate a sketch image $f_1(i, j)$. Specifically,

$$f_1(i,j) \leftarrow \text{round}(255 \times \frac{c(i,j)}{\max\{c(i,j)\}}), \tag{6.1}$$

where $c(i,j)$ denotes the number of non-zero AC coefficients in the (i,j)-th 8×8 block. Here, the number of non-zero AC coefficients infers the complexity of a block, a larger NCC implies higher spatial activity within the block, and vice versa.

2) EAC: The AC coefficients are exploited to generate a sketch image $f_2(i,j)$ as follows:

$$f_2(i,j) \leftarrow \text{round}(255 \times \frac{s(i,j)}{\bar{s}}), \tag{6.2}$$

where $s(i,j)$ denotes the sum of magnitude of AC coefficients in the (i,j)-th 8×8 block and

$$\bar{s} \leftarrow \frac{1}{(H \times W)/64} \sum_{i=1}^{H/8} \sum_{j=1}^{W/8} s(i,j), \tag{6.3}$$

H and W are the height and width of the image, respectively. Informally, $s(i,j)$ indicates the strength of the textures/edges, if any, in the (i,j)-th 8×8 block. Hence, larger $s(i,j)$ suggests stronger edges, and vice versa.

3) PLZ: The position (index with respect to the zigzag order) of the last non-zero AC coefficient is exploited to generate a sketch image $f_3(i,j)$ as follows:

$$f_3(i,j) \leftarrow \text{round}(255 \times \frac{p(i,j)}{\max\{p(i,j)\}}), \tag{6.4}$$

where $p(i,j)$ denotes the position of the last non-zero AC coefficient in the (i,j)-th 8×8 block. Larger PLZ implies more complex patter in the block, and vice versa.

In Figure 6.9, we take 'Baboon' image as example, show the sketch outline image produced by each of the above three methods, from which we can see that all three obtained sketch images disclose the outline information when plain 'Baboon' image is used for sketching. While for the two encrypted 'Baboon' images under *Algorithm-1-ch6* and *Algorithm-2-ch6*, no plain-image's

information can be deduced from the three sketch images, and this confirms that our proposed two encryption schemes can resist the three types of sketch attack.

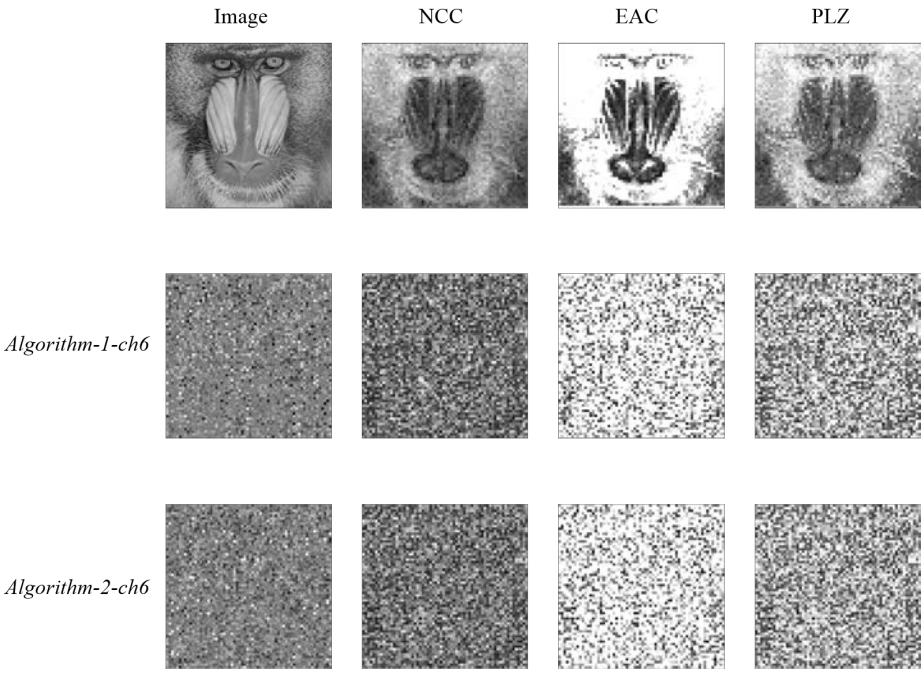


Figure 6.9: Sketch attack results of plain 'Baboon' image and encrypted 'Baboon' images.

6.4 Summary

In this chapter, we proposed a new idea to realize joint image compression and encryption by using adaptive block size for encryption. Encryption operations were mainly realized at JPEG's transformation stage and entropy coding stage. Based on different block segmentation criteria, two encryption schemes were developed. The main objective of the two methods was not to offer full confidentiality, but to provide compression with certain level of security robustness against various attacks, and not to compromise the compression performance.

For a given plain-image, we first raster scanned it into non-overlapping 16×16 blocks, and based on different segmentation criteria (key-based segmen-

tation or image-based segmentation), we divided each satisfied 16×16 block into four 8×8 block , and transformed these different size of blocks using corresponding DCT. Then we distributed 16×16 block's coefficients into 8×8 blocks, and shuffled all 8×8 blocks through Fisher-Yates shuffle. After quantizing all permuted 8×8 blocks, we obtained the RSV pairs for non-zero AC coefficients, and grouped them into several classes according to their run value. RSV pairs with the same run value were shuffled to realize AC coefficients encryption. Extensive experiments were conducted to show that the proposed two encryption schemes were both friendly to JPEG, meanwhile, they could provide a certain degree of protection for compressed images.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we first introduced some background of image encryption, like the differences existed between image data and text data, and criteria that were used for evaluating the performance of an image cryptosystem. Because nowadays most of the images we see on the Internet are compressed, and JPEG is the commonly used digital image compression standard, our current work mainly focused on protection of JPEG images. Four different JPEG image encryption schemes were developed, in which all encryption operations were implemented during JPEG's compression process, so as to complete image compression and encryption in one-step.

In Chapter 2, fundamentals of cryptography and cryptanalysis were presented, followed by a brief introduction of JPEG compression standard. Current existing JPEG image encryption algorithms were divided into three categories, according to the execution sequences of encryption and compression. These three categories were: pre-compression encryption algorithms, in-compression encryption algorithms, and post-compression encryption algorithms. Various algorithms in each category were described, together with

their advantages and disadvantages. In order to save the computation cost, we preferred to perform image encryption and image compression simultaneously, which belongs to the in-compression encryption category.

In Chapter 3, our first in-compression encryption scheme was presented, which was realized by modifying JPEG's transformation stage. We designed a new method for producing orthogonal transforms by introducing sign-flips into butterflies of 8×8 DCT's flow-graph structure. These newly produced transforms were then alternatively applied for 8×8 blocks' transformation, controlled by the secret key. Certain degree of image protection could be achieved under this cryptosystem, and quality control of the final encrypted images could also be realized through controlling the number of sign-flips introduced. Although the encryption power of this scheme was limited, its implementation was simple, could be easily integrated with other stages' encryption techniques, so that the encryption space could be enlarged by one dimension.

In Chapter 4, a content-adaptive in-compression encryption scheme was described, in which the secret key was generated from the plain-image using BLAKE2. The plain-image-dependent key mechanism could greatly enhance the cryptosystem's diffusion property, which was very important for resisting the differential attack. The whole scheme contained three encryption operations: alternating new orthogonal transforms transformation, DC coefficients encryption, and AC coefficients encryption. These new transforms were produced by allowing more rotation angles to be embedded into order-8 DCT's flow-graph, not just π (sign-flip), and the coding efficiency of these transforms was better than that of transforms produced in our first encryption scheme. The DC coefficients encryption was realized through 8×8 blocks' permutation and XOR operation, while AC coefficients were encrypted by embedding the plain-image produced secret key into bitstreams of some AC coefficients, and the embedding procedure was controlled by another secret key. Experimental

results have confirmed that this cryptosystem had good confusion and diffusion properties, could resist against many cryptanalysis methods, and maintained JPEG's compression performance well.

In Chapter 5, a 16×16 DCT-based joint image compression and encryption scheme was developed, and the results have shown that the cryptosystem's resistance ability against the statistical attack could be greatly improved by encrypting image in a basic unit of 16×16 data block. Two phases of encryption were involved in our proposed scheme. First, 16×16 coefficients transformed by order-16 DCT were re-distributed into 8×8 blocks, followed by 8×8 block permutation and DC coefficients confusion. In the second phase, encryption was incorporated in JPEG's entropy coding stage using RSV pairs shuffling and EOB identifiers embedding. The proposed scheme preserved the format of JPEG pipeline, and could obtain a good balance between the compression efficiency and encryption ability.

In Chapter 6, we realized JPEG image encryption by using adaptive block-size. Given a plain-image, we first segmented it into two sizes of blocks, 8×8 and 16×16, according to some segmentation criteria, then transformed these blocks by the corresponding size of DCT. After transformation, we distributed 16×16 transformed coefficients into 8×8 blocks, so as to keep accordance with JPEG's compression process. Then we permuted all 8×8 quantized blocks, and encrypted AC coefficients by shuffling those RSV pairs with same run value. Encryption ability of the proposed scheme was dependent on JPEG's quality factor, since large quality factor meant less compression, and more RSV pairs for shuffling, thus final encrypted images would be much more chaotic. Experimental results have illustrated that this scheme was JPEG compression friendly, had good defense ability against the statistical attack and sketch attack.

7.2 Future work

In the future, my research work can be summarized into the four main parts. First, various block sizes will be considered for selection in fixed and adaptive joint image compression and encryption schemes. Second, currently, the four proposed cryptosystems cannot achieve perfect correlation removal performance and good diffusion property, how to enhance these schemes' robustness against the differential attack and statistical attack will also be considered in the future. Additionally, the four encryption schemes proposed in this thesis are all worked for the JPEG standard, integration of these encryption techniques with other kind of compression standards, such as motion JPEG, H.264, and MPEG-4, will be taken into account. Finally, the application of our current proposed cryptosystems we believe can be further extended. For example, after encrypting and compressing digital images using our first encryption scheme, correlation existed in the encrypted DCT coefficients are not fully removed, they can still be explored to reflect features of the original unencrypted image, and this can be applied in privacy-preserving content-based image retrieval application.

References

[1] Y. Mao and M. Wu, "A joint signal processing and cryptographic approach to multimedia encryption," *IEEE Transactions on Image Processing*, vol. 15, no. 7, pp. 2061–2075, 2006.

[2] N. Kulkarni, B. Raman, and I. Gupta, "Multimedia encryption: a brief overview," *Recent advances in multimedia signal processing and communications*, pp. 417–449, 2009.

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

[4] J. Ahmad and F. Ahmed, "Efficiency analysis and security evaluation of image encryption schemes," *computing*, vol. 23, p. 25, 2010.

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

[6] A. Jolfaei, X.-W. Wu, and V. Muthukkumarasamy, "A secure lightweight texture encryption scheme," in *Pacific-Rim Symposium on Image and Video Technology*. Springer, 2015, pp. 344–356.

[7] M. A. F. Al-Husainy, "A novel encryption method for image security," *International Journal of Security and Its Applications*, vol. 6, no. 1, 2012.

[8] A. Uhl and A. Pommer, *Image and video encryption: from digital rights management to secured personal communication*. Springer Science & Business Media, 2004, vol. 15.

[9] S.-K. A. Yeung, S. Zhu, and B. Zeng, "Quality assessment for a perceptual video encryption system," in *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on*. IEEE, 2010, pp. 102–106.

[10] H. Hofbauer and A. Uhl, "Identifying deficits of visual security metrics for images," *Signal Processing: Image Communication*, vol. 46, pp. 60–75, 2016.

[11] B. M. Macq and J.-J. Quisquater, "Cryptology for digital tv broadcasting," *Proceedings of the IEEE*, vol. 83, no. 6, pp. 944–957, 1995.

[12] Z. Wang and A. C. Bovik, "A universal image quality index," *Signal Processing Letters, IEEE*, vol. 9, no. 3, pp. 81–84, 2002.

[13] J. Wen, M. Severa, W. Zeng, M. H. Luttrell, and W. Jin, "A format-compliant configurable encryption framework for access control of video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 545–557, 2002.

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

[15] 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.

[16] Y. Xu, L. Xiong, Z. Xu, and S. Pan, "A content security protection scheme in jpeg compressed domain," *Journal of Visual Communication and Image Representation*, vol. 25, no. 5, pp. 805–813, 2014.

[17] R. Kaur and E. K. Singh, "Image encryption techniques: A selected review," *Journal of Computer Engineering (IOSRJCE)*, vol. 9, no. 6, pp. 80–83, 2013.

[18] B. Furht and D. Kirovski, *Multimedia security handbook*. CRC press, 2004.

[19] M. Yang, N. Bourbakis, and S. Li, "Data-image-video encryption," *IEEE potentials*, vol. 23, no. 3, pp. 28–34, 2004.

[20] W. Stallings and M. P. Tahiliani, *Cryptography and network security: principles and practice*. Pearson London, 2014, vol. 6.

[21] H. Cheng and X. Li, "Partial encryption of compressed images and videos," *IEEE Transactions on signal processing*, vol. 48, no. 8, pp. 2439–2451, 2000.

[22] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[23] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *IEEE Signal processing magazine*, vol. 18, no. 5, pp. 36–58, 2001.

[24] "ISO/IEC JTC 1/SC 29/WG 1, JPEG Privacy and Security Call for Proposals," The JPEG Committee, 2017, available at https://jpeg.org/downloads/privacy_and_security/wg1n74015_Draft_CfP.pdf.

[25] K. S. Thyagarajan, *Still image and video compression with MATLAB*. John Wiley & Sons, 2011.

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

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

[28] 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.

[29] 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.

[30] Y. Zhang and D. Xiao, "An image encryption scheme based on rotation matrix bit-level permutation and block diffusion," *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, no. 1, pp. 74–82, 2014.

[31] 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.

[32] 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.

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

[34] C. Fu, B.-b. Lin, Y.-s. Miao, X. Liu, and J.-j. Chen, "A novel chaos-based bit-level permutation scheme for digital image encryption," *Optics communications*, vol. 284, no. 23, pp. 5415–5423, 2011.

[35] G. Ye, "Image scrambling encryption algorithm of pixel bit based on chaos map," *Pattern Recognition Letters*, vol. 31, no. 5, pp. 347–354, 2010.

[36] J. Hu and F. Han, "A pixel-based scrambling scheme for digital medical images protection," *Journal of Network and Computer Applications*, vol. 32, no. 4, pp. 788–794, 2009.

[37] D. Ponnain and K. Chandranbabu, "Crypt analysis of an image encryption algorithm and an enhanced scheme," *Optik-International Journal for Light and Electron Optics*, vol. 127, no. 1, pp. 192–199, 2016.

[38] M. Kumar, P. Powduri, and A. Reddy, "An rgb image encryption using diffusion process associated with chaotic map," *Journal of Information Security and Applications*, vol. 21, pp. 20–30, 2015.

[39] O. S. Faragallah, "Efficient confusion–diffusion chaotic image cryptosystem using enhanced standard map," *Signal, Image and Video Processing*, vol. 9, no. 8, pp. 1917–1926, 2015.

[40] R. Guesmi, M. A. B. Farah, A. Kachouri, and M. Samet, "Hash key-based image encryption using crossover operator and chaos," *Multimedia tools and applications*, vol. 75, no. 8, pp. 4753–4769, 2016.

[41] A. Kanso and M. Ghebleh, "A novel image encryption algorithm based on a 3d chaotic map," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 7, pp. 2943–2959, 2012.

[42] J. Xie, C. Yang, Q. Xie, and L. Tian, "An encryption algorithm based on transformed logistic map," in *Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC'09. International Conference on*, vol. 2. IEEE, 2009, pp. 111–114.

[43] H. Gao, Y. Zhang, S. Liang, and D. Li, "A new chaotic algorithm for image encryption," *Chaos, Solitons & Fractals*, vol. 29, no. 2, pp. 393–399, 2006.

[44] 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.

[45] 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.

[46] 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.

[47] W. Li, Y. Yan, and N. Yu, "Breaking row-column shuffle based image cipher," in *Proceedings of the 20th ACM international conference on Multimedia*. ACM, 2012, pp. 1097–1100.

[48] S. Li, C. Li, K.-T. Lo, and G. Chen, "Cryptanalysis of an image scrambling scheme without bandwidth expansion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 3, pp. 338–349, 2008.

[49] N. K. Pareek, V. Patidar, and K. K. Sud, "Diffusion–substitution based gray image encryption scheme," *Digital signal processing*, vol. 23, no. 3, pp. 894–901, 2013.

[50] A. Jolfaei, X.-W. Wu, and V. Muthukkumarasamy, "Comments on the security of diffusionsubstitution based gray image encryption scheme," *Digital Signal Processing*, vol. 32, no. Supplement C, pp. 34 – 36, 2014.

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

[52] K. Kurihara, M. Kikuchi, S. Imaizumi, S. Shiota, and H. Kiya, "An encryption-then-compression system for jpeg/motion jpeg standard," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 98, no. 11, pp. 2238–2245, 2015.

[53] 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.

[54] X. Zhang, Y. Ren, L. Shen, Z. Qian, and G. Feng, "Compressing encrypted images with auxiliary information," *IEEE transactions on multimedia*, vol. 16, no. 5, pp. 1327–1336, 2014.

[55] X. Zhang, G. Feng, Y. Ren, and Z. Qian, "Scalable coding of encrypted images," *IEEE Transactions on Image Processing*, vol. 21, no. 6, pp. 3108–3114, 2012.

[56] X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE transactions on information forensics and security*, vol. 7, no. 2, pp. 826–832, 2012.

[57] ——, "Lossy compression and iterative reconstruction for encrypted image," *IEEE transactions on information forensics and security*, vol. 6, no. 1, pp. 53–58, 2011.

[58] B. Zeng, S.-K. A. Yeung, S. Zhu, and M. Gabbouj, "Perceptual encryption of h. 264 videos: Embedding sign-flips into the integer-based transforms," *Information Forensics and Security, IEEE Transactions on*, vol. 9, no. 2, pp. 309–320, 2014.

[59] S.-K. A. Yeung, S. Zhu, and 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.

[60] ——, "Design of new unitary transforms for perceptual video encryption," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 9, pp. 1341–1345, 2011.

[61] ——, "Partial video encryption based on alternative integer transforms," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on.* IEEE, 2010, pp. 933–936.

[62] ——, "Partial video encryption based on alternating transforms," *Signal Processing Letters, IEEE*, vol. 16, no. 10, pp. 893–896, 2009.

[63] C. Shi and B. Bhargava, "A fast mpeg video encryption algorithm," in *Proceedings of the sixth ACM international conference on Multimedia.* ACM, 1998, pp. 81–88.

[64] W. Zeng and S. Lei, "Efficient frequency domain selective scrambling of digital video," *IEEE Transactions on Multimedia*, vol. 5, no. 1, pp. 118–129, 2003.

[65] P. Korshunov and T. Ebrahimi, "Scrambling-based tool for secure protection of jpeg images," in *Image Processing (ICIP), 2014 IEEE International Conference on.* IEEE, 2014, pp. 3423–3425.

[66] W. Li and 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.

[67] L. Krikor, S. Baba, T. Arif, and Z. Shaaban, "Image encryption using dct and stream cipher," *European Journal of Scientific Research*, vol. 32, no. 1, pp. 47–57, 2009.

[68] C.-P. Wu and C.-C. J. Kuo, "Fast encryption methods for audiovisual data confidentiality," in *Proceedings of SPIE, the International Society for Optical Engineering*, vol. 4209. Society of Photo-Optical Instrumentation Engineers, 2001, pp. 284–295.

[69] H. Hofbauer, A. Unterweger, and A. Uhl, "Encrypting only ac coefficient signs considered harmful," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3740–3744.

[70] 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.

[71] 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.

[72] K. Wong and K. Tanaka, "Dct based scalable scrambling method with reversible data hiding functionality," in *Communications, Control and Signal Processing (ISCCSP), 2010 4th International Symposium on*. IEEE, 2010, pp. 1–4.

[73] 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.

[74] Y. Lu, W. Yang, and L. Chen, "Encryption algorithm for the image in the frequency domain," *Computer Engineering and Application*, vol. 39, no. 14, pp. 130–131, 2003.

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

[76] L. Qiao, K. Nahrstedt, and 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.

[77] S. U. Shin, K. S. Sim, and K. H. Rhee, "A secrecy scheme for mpeg video data using the joint of compression and encryption," in *International Workshop on Information Security.* Springer, 1999, pp. 191–201.

[78] S. S. Maniccam and N. G. Bourbakis, "Image and video encryption using scan patterns," *Pattern Recognition*, vol. 37, no. 4, pp. 725–737, 2004.

[79] S. Jha, "Image compresssion and encryption using scan pattern," Ph.D. dissertation, 2014.

[80] X.-y. Ji, S. Bai, Y. Guo, and H. Guo, "A new security solution to jpeg using hyper-chaotic system and modified zigzag scan coding," *Communications in Nonlinear Science and Numerical Simulation*, vol. 22, no. 1, pp. 321–333, 2015.

[81] C. Kailasanathan and R. S. Naini, "Compression performance of jpeg encryption scheme," in *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, vol. 2. IEEE, 2002, pp. 1329–1332.

[82] L. Qiao and K. Nahrstedt, "Comparison of mpeg encryption algorithms," *Computers & Graphics*, vol. 22, no. 4, pp. 437–448, 1998.

[83] T. Uehara and R. Safavi-Naini, "Chosen dct coefficients attack on mpeg encryption schemes," in *Proceedings of IEEE Pacific Rim Conference on Multimedia*, 2000, pp. 316–319.

[84] 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.

[85] C.-P. Wu and C.-C. J. Kuo, "Efficient multimedia encryption via entropy codec design," in *Proc. SPIE*, vol. 4314, 2001, pp. 128–138.

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

[87] 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.

[88] A. Unterweger and A. Uhl, "Length-preserving bit-stream-based jpeg encryption," in *Proceedings of the on Multimedia and security.* ACM, 2012, pp. 85–90.

[89] X. Niu, C. Zhou, J. Ding, and B. Yang, "Jpeg encryption with file size preservation," in *Intelligent Information Hiding and Multimedia Signal Processing, 2008. IIHMSP'08 International Conference on.* IEEE, 2008, pp. 308–311.

[90] D. Xie and C.-C. Kuo, "Multimedia data encryption via random rotation in partitioned bit streams," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on.* IEEE, 2005, pp. 5533–5536.

[91] 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.

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

[93] D. Socek, H. Kalva, S. S. Magliveras, O. Marques, D. Culibrk, and B. Furht, "New approaches to encryption and steganography for digital videos," *Multimedia Systems*, vol. 13, no. 3, pp. 191–204, 2007.

[94] D. Zhang and F. Zhang, "Chaotic encryption and decryption of jpeg image," *Optik-International Journal for Light and Electron Optics*, vol. 125, no. 2, pp. 717–720, 2014.

[95] 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.

[96] 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.

[97] K. Kaukonen and R. Thayer, "A stream cipher encryption algorithm arcfour," 1999.

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

[99] L. Bao, Y. Zhou, C. P. Chen, and H. Liu, "A new chaotic system for image encryption," in *System Science and Engineering (ICSSE), 2012 International Conference on*. IEEE, 2012, pp. 69–73.

[100] A. A. Bruen, M. A. Forcinito, A. G. Konheim, C. Cobb, A. Young, M. Yung, and D. Hook, "Applied cryptography: protocols, algorithms, and source code in c," 1996.

[101] 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.

[102] S. S. Agaian, R. G. Rudraraju, and 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.

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

[104] K. Pearson, "Contributions to the mathematical theory of evolution," *Philosophical Transactions of the Royal Society of London. A*, vol. 185, pp. 71–110, 1894.

[105] 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.

[106] 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.

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

[108] 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.

[109] 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.

[110] G. Schaefer and M. Stich, "Ucid: An uncompressed color image database," in *Storage and Retrieval Methods and Applications for Multimedia 2004*, vol. 5307. International Society for Optics and Photonics, 2003, pp. 472–481.

[111] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on.* IEEE, 1998, pp. 200–205.

[112] 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.

[113] K. Minemura, Z. Moayed, K. Wong, X. Qi, and K. Tanaka, "Jpeg image scrambling without expansion in bitstream size," in *Image Processing (ICIP), 2012 19th IEEE International Conference on.* IEEE, 2012, pp. 261–264.

[114] K. Minemura, K. Wong, X. Qi, and K. Tanaka, "A scrambling framework for block transform compressed image," *Multimedia Tools and Applications*, vol. 76, no. 5, pp. 6709–6729, 2017.

[115] S. Ong, K. Minemura, and K. Wong, "Progressive quality degradation in jpeg compressed image using dc block orientation with rewritable data embedding functionality," in *Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE, 2013, pp. 4574–4578.

[116] 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.

[117] S. Xu, Y. Wang, J. Wang, and Y. Guo, "A fast image encryption scheme based on a nonlinear chaotic map," in *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, vol. 2. IEEE, 2010, pp. V2–326.

[118] L. Zhang, X. Liao, and X. Wang, "An image encryption approach based on chaotic maps," *Chaos, Solitons & Fractals*, vol. 24, no. 3, pp. 759–765, 2005.

[119] K. Loukhaoukha, J.-Y. Chouinard, and A. Berdai, "A secure image encryption algorithm based on rubik's cube principle," *Journal of Electrical and Computer Engineering*, vol. 2012, p. 7, 2012.

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

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

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

[123] J. Zhou, X. Liu, O. C. Au, and Y. Y. Tang, "Designing an efficient image encryption-then-compression system via prediction error clustering and random permutation," *IEEE transactions on information forensics and security*, vol. 9, no. 1, pp. 39–50, 2014.

[124] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, "Raise: a raw images dataset for digital image forensics," in *Proceedings of the 6th ACM Multimedia Systems Conference*. ACM, 2015, pp. 219–224.

[125] E. Shusterman and M. Feder, "Image compression via improved quadtree decomposition algorithms," *IEEE Transactions on image processing*, vol. 3, no. 2, pp. 207–215, 1994.