

Encryption and Decryption Methods and Apparatus

BACKGROUND OF THE INVENTION

5

1. Field of the Invention

The present invention relates to the encryption and decryption of image data, such as of JPEG encoded images.

10 2. Description of the Related Art

JPEG is a standardized image compression mechanism that is designed for compressing full-color or gray scale images of natural, real-world scenes. The name JPEG stands for Joint Photographic Experts Group, the name of the joint ISO/CCITT committee that created the standard. The group was organized in 1986, issuing a
15 standard in 1992 that was approved in 1994 as ISO 10918-1. JPEG itself specifies only how an image is transformed into a stream of bytes, but not how those bytes are encapsulated in any particular storage medium. A further standard, created by the Independent JPEG Group, called JFIF (JPEG File Interchange Format), specifies how to produce a file from a JPEG stream that is suitable for computer storage and
20 transmission (such as over the Internet). In common usage, when one speaks of a "JPEG file" one generally means a JFIF file. JPEG is used for lossy compression of images, although, variations on the standard baseline JPEG allow for lossless compression. The most common file extensions for this format are .jpeg and .jpg.

Today JPEG is one of the most widely used formats by digital cameras for
25 storing images. This format is also commonly used for images on the web and images attached to e-mail messages. It is also quite popular for digital security cameras that provide data streaming through networks.

Due to the success of more powerful personal computers and high-speed Internet connections, more and more photos that are captured in JPEG format are
30 shared on the Internet among different groups of people, e.g., friends, relatives, colleagues, etc. Moreover, live streaming of JPEG images from security cameras for surveillance through the Internet is more and more popular. Conventionally, these JPEG images are not protected by cryptography algorithms when transmitted through the Internet. Unfortunately, however, Internet eavesdropping, hacking, piracy, and
35 illegal global distribution of digital content are big problems, and content owners are always concerned about protecting their copyright works from illegal use and from

unauthorized users, especially security-sensitive images (e.g. private photos and images from security cameras). Therefore, in order to reduce the risk of hacking and piracy, JPEG images should be protected when transmitted through the Internet. In addition, applying cryptography algorithms on JPEG images during storage would help protect those images from loss and stealing.

There are some companies that provide encryption on JPEG images, e.g. for security cameras. For instance, Milestone Systems (<http://www.milestonesys.com>) provides a solution for IP video (JPEG) surveillance by scrambling the images with a password during transmission and exporting. This can provide security, but requires the use of dedicated software to descramble protected images before they may be analyzed, and no other software can analyze the encrypted JPEG images.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide an encryption/decryption solution to protect image files, such as JPEG image files. The encryption process may protect files during transmission on the Internet as well when stored on digital recording media (e.g. SD memory cards, hard disks, CD/DVD discs, etc.).

Viewed from one aspect, the present invention provides a method of encrypting an image file of a digital image, the method including the steps of:

identifying marker codes in the image file corresponding to file data;

identifying payload data in the image file corresponding to image data;

applying an encryption process to the payload data to provide encrypted payload data; and

constructing an encrypted image file from unencrypted data associated with the marker codes and from the encrypted payload data.

By encrypting payload data but retaining various marker codes and associated data as plaintext, the encryption method enables the image file to be securely prepared, delivered and consumed, e.g. through the Internet or packaged media, and to be securely stored, whilst also enabling the encrypted file to be scanned by standard image file handling applications, e.g. JPEG decoders and the like, to obtain file data. It can prevent the reading of actual image data that would allow the digital image to be reconstructed, but enables applications to obtain file data, e.g. information regarding the image file and/or information regarding the digital image without requiring decryption of the file.

For example, with a JPEG image only the first JPEG scan payload data and

data following it until the last EOI marker code may be encrypted, and marker codes and associated data before this may remain as plaintext. These marker codes may include various file headers and the like, and, for example, marker codes and data, such as SOI, EOI, Frame Header, first Scan Header, and the like, may remain as
5 plaintext. When Frame Header and Scan Header information are not encrypted, any image decoder, e.g. a JPEG decoder, can analyze the image properties, such as image size and coding method. Scan Data however is not necessarily available, so that the image cannot be decoded (from e.g. the JPEG encoding) until it is decrypted.

Preferably, the scrambled image data is of the same size as the unencrypted
10 image data, so that applications processing the image files do not generate error conditions through differences between the data sizes of the existing ciphertext and the original data sizes of the plaintext image data recorded in the retained plaintext headers. A number of methods for achieving this are discussed below. They may generally be termed Equal Payload Length Encryption and Decryption (EPL
15 mechanisms, and may be implemented using suitable EPLE Scramblers and Descramblers.

The EPLE processes may for example use block cipher algorithms, and, in one embodiment, a single key may be used to encode a maximum amount of payload data that is a multiple of a block size of the encryption process, with the remaining
20 small amount of payload data being left unencrypted. In another embodiment, two keys are used with one used to encrypt an amount of data corresponding to a single block size of the encryption process and the other used to encrypt an amount of data corresponding to a multiple of the block size. In a further embodiment, each key is used to encrypt an amount of data corresponding to a multiple of the block size. In
25 both situations, there will generally be an overlap of encryption, so that some parts of the payload data are encrypted by both keys.

Preferably, the encryption processes scan for data values in the generated ciphertext that accidentally correspond to marker codes, and replace them with dummy data. This can prevent applications that scan the encrypted data from
30 determining that a portion of the ciphertext is an actual marker code, and from causing an error through an attempt to act in accordance with the marker code. The method may for example replace a portion of the marker codes that is common to all marker codes (e.g. FF) with neutral dummy data (e.g. 00).

The location of the replaced marker code values may be stored at the end of
35 the encrypted image file or in a separate file, and may be used to reconstruct the ciphertext prior to decryption.

The replacement of such marker codes in the ciphertext may only be necessary if a JPEG decoder or the like has to scan through the ciphertext, e.g. when a JPEG decoder is to produce a mosaic, that is display a picture formed by decoding the encrypted image data without decrypting it first, so that the image displayed consists of randomly coloured blocks having no recognizable patterns.

The image file may for example be generated by any JPEG encoder, as long as the encoder is in accordance with the JPEG standard and the JFIF standard.

The method may allow for the setting of an Encryption Strategy for initialization of an EPLE Scrambler and an EPLE Descrambler, which may define for example: a Block Cipher Algorithm, e.g. DES, AES, Blowfish, etc; seeds for generating random keys; preferred key lengths for Block Cipher Algorithms that accept variable length keys; and preferred Plaintext Block Sizes, e.g. 64-bit, 128-bit, 256-bit, etc. A default Encryption Strategy may also be provided.

The present invention also extends to decryption methods for decrypting image files that have been encrypted using the present encryption processes, and extends to systems that implement the encryption and decryption processes. It further extends to software products including code for providing the encryption and decryption processes, and to image files produced using the encryption and decryption processes.

Although discussed mainly in the context of JPEG images, the present invention also has broader application to any suitable type of image file, e.g. those having headers and payloads, including for example image files in the bitmap file format, where the same principles may be applied, with header information retained as plaintext and with payload data that corresponds to the encoded image as ciphertext.

The nature, principles and utility of the invention will become more apparent from the following detailed description, which is given by way of example only, when read in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

In the accompanying drawings:

Figure 1 is a diagram showing a system architecture of a JPEG Image Encryption System;

Figure 2 is a diagram showing a system architecture of a JPEG

Image Decryption System;

Figure 3 is a diagram showing a JPEG Structure and data that may be scrambled under an EPLE process;

Figure 4 is a diagram showing a JFIF Structure;

5 Figure 5 is a diagram showing headers and payload defined for a JPEG image in an EPLE mechanism;

Figures 6(a) to 6(e) are diagrams showing possible encryption/decryption processes for payload data in an EPLE mechanism;

10 Figure 7 is a diagram showing another encryption process for payload data in an EPLE mechanism;

Figure 8 is a diagram showing another encryption process for payload data in an EPLE mechanism;

15 Figure 9 is a diagram showing a decryption process for the encryption process of Figure 7;

Figure 10 is a diagram showing a decryption process for the encryption process of Figure 8;

Figure 11 is a diagram showing a further encryption process for payload data in an EPLE mechanism;

20 Figure 12 is a diagram showing a decryption process for the encryption process of Figure 11;

Figure 13 is a diagram showing a further encryption process for payload data in an EPLE mechanism;

25 Figure 14 is a diagram showing a decryption process for the encryption process of Figure 13; and

Figure 15 is a diagram showing a file structure for Unexpected Marker Code Indices;

DESCRIPTION OF THE PREFERRED EMBODIMENTS

30

Shown in Figure 1 and Figure 2 are embodiments of a JPEG image (in accordance with JFIF format) encryption and decryption system, wherein a JPEG image is securely protected.

35 A JPEG Image Loader, a Data Buffer, an EPLE Scrambler, an EPLE Descrambler, a Data Post-processor, and a Data Pre-processor may be implemented in any suitable information-processing devices that can execute the processes

described below. These information-processing devices may include general-purpose computers, workstations, and personal computers, as well as network connectable information-processing devices, for instance digital home electric appliances, and portable terminals, for instance PDAs and cellular phones. It should be noted that the processes described below may be performed by a software product and/or may be performed in hardware.

Referring to Figures 1 and 2, the system may comprise:

- (1) A JPEG Image Loader which:
 - (a) Loads a JPEG image/file, which is in accordance with JFIF format, to a Data Buffer in a memory, or,
 - (b) Loads consecutive JPEG images/files to a Data Buffer in a memory according to a corresponding Time Stamp, or,
 - (c) Loads JPEG images/files to a Data Buffer in a memory according to predefined Timers.
- (2) A JPEG Image which may be:
 - (a) A JPEG bytes stream encapsulated by JFIF for computer storage and/or transmission over a network.
- (3) A Data Buffer which:
 - (a) Is a block of memory assigned for storage of JPEG image data,
 - (b) Is the location to store encrypted JPEG image data.
- (4) An EPLE Scrambler which
 - (a) Implements an EPLE mechanism to scramble JPEG image data stored in the Data Buffer,
 - (b) Passes parameters such as Ciphertext Location and Ciphertext Length to a Data Post-processor.
- (5) A Data Post-processor which
 - (a) Analyzes encrypted JPEG image data,
 - (b) Locates Unexpected Marker Codes in the ciphertext,
 - (c) Replaces Unexpected Marker Codes with normal data if necessary
 - (d) Generates Unexpected Marker Code Indices if necessary,
 - (e) Attaches Unexpected Marker Code Indices to the end of a JPEG image file if necessary.
- (6) A Data Pre-processor which
 - (a) Analyzes Unexpected Marker Code Indices if necessary,
 - (b) Finds the location of Unexpected Marker Codes in the ciphertext based on the Unexpected Marker Code Indices if

- necessary,
 - (c) Restores Unexpected Marker Codes if necessary,
 - (d) Removes Unexpected Marker Code Indices from the end of the JPEG image if necessary.
- 5 (7) An EPLE Descrambler which
- (a) Implements an EPLE mechanism to descramble JPEG image data stored in a Data Buffer.

10 Figure 1 and Figure 2 generally show the architecture of a JPEG image encryption/decryption system that may implement the present invention. In a preferred embodiment of the present invention, a JPEG image/file could be generated by any JPEG encoder, as long as it is in accordance with the JPEG standard and the JFIF standard.

15 The embodiment allows an Encryption Strategy to be defined for initialization of an EPLE Scrambler and Descrambler, which may define: a Block Cipher Algorithm, e.g. DES, AES, Blowfish, etc; seeds for generating random Keys; preferred Key Length for Block Cipher Algorithms that accept variable length keys; and preferred Plaintext Block Sizes, e.g. 64-bit, 128-bit, 256-bit, etc. Otherwise, a default Encryption Strategy may be

20 used.

 In a preferred embodiment, the scrambled JPEG image maintains the same data size, and this is implemented by the EPLE Scrambler and the EPLE Descrambler which employ an encryption process that produces an encrypted payload that is of the same length as the unencrypted payload, and which may implement

25 Block Cipher Algorithms to encrypt data. The EPLE mechanism may encrypt/decrypt only the first JPEG Scan payload data and the data following it, and headers and other marker codes that occur before the first Scan payload data may be left untouched. For instance, SOI, EOI, Frame Header, the first Scan Header, and the like may remain untouched.

30 Figure 3 shows the JPEG Structure and data that may be scrambled under an EPLE mechanism. Figure 4 shows the JFIF structure.

 Since Frame Header and Scan Header information are not encrypted, the scrambled JPEG image is available for any JPEG Decoder to analyze regarding image properties, such as image size, coding method, etc, but the scan data is not

35 available for decoding before being correctly decrypted. Therefore, information regarding the image may be obtained, but the image itself is not accessible.

In a preferred embodiment of the present invention, the Data Post-processor analyzes the encrypted JPEG image data, and locates Unexpected Marker Codes in the ciphertext. Unexpected Marker Codes are data values produced in the encryption process that correspond with recognized JPEG Marker Codes. Their retention in the
5 ciphertext may cause problems if they are read by a JPEG decoder as actual Marker Codes. For example, Unexpected Marker Codes may be regarded as a decoding termination signal by a JPEG decoder. Although not necessary, a JPEG decoder may be allowed to decode the protected JPEG image without decryption, for example so that mosaics may be shown (mosaics are an image of randomly coloured blocks that
10 do not have any recognizable patterns and that are formed from the decoding of image data that has not been decrypted). In this case, the Unexpected Marker Codes are preferably replaced with normal data, e.g. dummy data of a neutral form. The Data Post-processor can replace Unexpected Marker Codes with normal data and may generate Unexpected Marker Code Indices that indicate the location of the
15 replacements. The Unexpected Marker Code Indices can be attached to the end of the JPEG image according to a predefined data structure. The Unexpected Marker Code Indices can alternatively be saved as an additional file. The embodiment allows a Data Pre-processor to analyze the Unexpected Marker Code Indices to restore Unexpected Marker Codes before decryption if Unexpected Marker Codes in the
20 ciphertext have been replaced.

In order to keep the integrity and consistency of a JPEG image, the Encryption/Decryption should firstly not introduce redundant bits into the content after it is encrypted. Additional bits can mean great modifications to the original JPEG data. Secondly, the complexity of the file systems and network protocols need
25 to be considered. Headers and Marker Codes, for instance, SOI, EOI, Frame Header, the first Scan Header, etc, may be used during network transfer or content management and analysis. By only applying encryption/decryption on the scan payload data and keeping most headers untouched, the encrypted JPEG image can still be manipulated by existing file systems and network protocols without any
30 additional components or modifications on those protocols. Furthermore, protection of the payload data, which contains the real image information, is enough to prevent

illegal access to the image because with only header information, which may just describe the payload data, nothing can be decoded. Other considerations are that the encryption/decryption algorithm be sufficiently fast, easy to implement, and good “value-of-money”. The EPLE mechanisms discussed herein are able to meet these targets.

Various encryption procedures are discussed below. They assume that a predefined Encryption Strategy has been selected, which may define: the Block Cipher Algorithms, e.g. DES, AES, Blowfish, etc; seeds for generating random Keys; preferred Key Lengths for the Block Cipher Algorithms that accept variable length keys; and preferred Plaintext Block Sizes, e.g. 64-bit, 128-bit, 256-bit, etc. In the following, the Plaintext Block Size is assumed to be N_b bits, the Key Length N_k bits, the Header Length N_h bits, and the Payload Length N_p bits. Figure 5 shows the headers and the payload for a JPEG image in an Equal Payload Length Encryption mechanism.

A simple solution for EPLE to protect a JPEG image is to directly apply Block Cipher Algorithms on the payload data. According to the requirement of a Block Cipher Algorithm, the total length of the plaintext should be a multiple of N_b . But most of the time, it is not true for N_p . Assuming the key is K with length N_k , there are two different cases, i.e., $N_p \geq N_b$ and $N_p < N_b$. The following operations may be carried out:

$$(1) \quad N_p \geq N_b$$

a. Do modulus calculation,

$$n = N_p \bmod N_b$$

$$\text{and, } z = (N_p - n) / N_b$$

where, z is an integer value and n is the remainder of the modulus calculation.

b. $n \in [0, N_b)$. So we can skip the first n bits of payload data in the buffer, encrypt the remaining zN_b bits of payload data in the buffer with K , and replace the corresponding plaintext with the ciphertext, which is shown in Figure 6(a).

- c. Alternatively, we can encrypt the first zN_b bits of payload data in the buffer with K, skip the last n bits of payload data, and replace the corresponding plaintext with the ciphertext, which is shown in Figure 6(b).

5

Before the protected JPEG image can be decoded and rendered, it must be decrypted. The decryption process is just the reverse process of the encryption process. Decryption of a JPEG image will use the same strategy that is defined in the encryption mechanism. For Block Cipher Algorithms, which are also called

10 Symmetric Encryption Algorithms, the Decryption Keys are the same keys as Encryption Keys. The corresponding decryption processes are shown in the following:

$$(2) \quad N_p \geq N_b$$

- a. Do modulus calculation,

$$15 \quad n = N_p \bmod N_b$$

$$\text{and, } z = (N_p - n) / N_b$$

where, z is an integer value and n is the remainder of the modulus calculation.

- b. $n \in [0, N_b)$. So we can skip the first n bits of payload data in the buffer,

20 decrypt the remaining zN_b bits of payload data in the buffer with K, and replace the corresponding ciphertext with the plaintext, which is shown in Figure 6(c).
- c. Alternatively, if the first zN_b bits of payload data are encrypted, we can

25 decrypt the first zN_b bits of payload data in the buffer with K, skip the last n bits of payload data, and replace the corresponding ciphertext with the plaintext, which is shown in Figure 6(d).

$$(3) \quad N_p < N_b$$

Such a case seldom happens because N_p is unlikely to be less than 64-bit, or

30 128-bit, or even 256-bit, which are normal key lengths. We just skip all of the

payload data for such a case, i.e. there is no encryption/decryption of the payload data, which is shown in Figure 6(e).

In the above mechanism, some data (n bits) will not be protected. Often, this will not be important, but sometimes the n bits of data may have important information. In the following, we introduce further EPLE mechanisms to address this, in which two keys are applied to the payload data at the same time, which are presented as (K_1, K_2) . Several ways of applying a two key encryption process are discussed.

$$(4) \quad N_p \geq N_b$$

The detail of the algorithm is shown in the following, and is also shown in Figure 7.

- a. Each JPEG image is loaded to the Buffer and analyzed to classify the Header and the Payload as defined in Figure 5,
- b. Compare N_p and N_b . If $N_p \geq N_b$, encrypt the first N_b bits of the payload data with K_1 , then replace the plaintext with the corresponding ciphertext in the Buffer,
- c. Do modulus calculation,

$$n = N_p \bmod N_b$$

$$\text{and, } z = (N_p - n) / N_b$$

where, z is an integer value and n is the remainder of the modulus calculation,

- d. $n \in [0, N_b)$, and we skip the first n bits in the Buffer, which is part of the ciphertext from the above steps that is encrypted with K_1 ,
- e. Encrypt the remaining zN_b bits in the Buffer with K_2 , and replace the plaintext with the ciphertext,
- f. The JPEG image data is re-constructed by combining the original Header and the encrypted payload data without any additional bits,
- g. The scrambled JPEG image data is fed to the Data Post-processor, if necessary. The process of Data Post-processor will be described later.

The extreme case is that $n = 0$. Then the first N_b bits of the payload data will be encrypted with K_1 and K_2 . However, as $K_1 \neq K_2$, the first N_b bits is not recovered but encrypted twice.

5

$$(5) \quad N_p < N_b$$

Because normally N_b is 64 (e.g. DES), or 128 (e.g. AES), or 256 (e.g. AES), this situation seldom occurs, as JPEG images do not normally occur for an N_p that is only 8, or 16, or 32 bytes. For such a case, we implement a Cipher Feedback (CFB) operation mode of Block Cipher Algorithms. In this mode, the plaintext itself is not actually passed through the Block Cipher Algorithms, but is merely XORed with an output block from it, which is shown in the following manner,

10

15

20

- a. An N_b bits block called the Shift Register, which is initially set to some arbitrary value, is used as the input plaintext to the Block Cipher Algorithm,
- b. The ciphertext is then passed through an extra component called the M-Box, which simply selects the left-most M bits of the ciphertext, where M is the number of bits in the data block that we wish to encrypt, i.e., N_p ,
- c. The selected M -bit is XORed with the real plaintext that we wish to encrypt,
- d. The output of the XOR operation is the final ciphertext.

25

30

In our system, we use K_1 as the initial Shift Register value, V_r . If the length of the keys, N_k , is less than N_b , we simply add several dummy bits to the end of K_1 , which may be set to zero, to make V_r the same length as N_b . If N_k is greater than N_b , we take only the left-most N_b bits from K_1 as the initial Shift Register value. After that, we encrypt V_r with K_2 getting the ciphertext of V_c . Then the left-most N_p bits of V_c are selected to XOR the N_p bits of payload data. The result is the final ciphertext. The procedure is shown in Figure 8.

In the following, we show how the corresponding decryption procedures are implemented. We assume that a predefined Encryption/Decryption Strategy has been selected, which may define: the Block Cipher Algorithm, e.g. DES, AES, Blowfish, etc; seeds for generating random Keys; preferred Key Lengths for the Block Cipher Algorithms that accept variable length keys; and preferred Plaintext Block Sizes, e.g., 64-bit, 128-bit, 256-bit, etc. At first, we again assume the Plaintext Block Size is N_b bits, the Key Length is N_k bits, the Header Length is N_h bits, and the Payload Length is N_p bits. Two keys will be applied to the payload at the same time, which is presented as (K_1, K_2) . For decryption, the same situations as for encryption have to be considered.

$$(6) \quad N_p \geq N_b$$

According to our encryption mechanism, the original header information will be untouched after encryption. With (K_1, K_2) the detail of the decryption algorithm is shown in the following, and also in Figure 9.

- a. Each protected JPEG image will be loaded to the Buffer and processed by the Data Pre-processor that will be described later, if necessary. Then it will be analyzed to classify the Header and the Payload as defined in Figure 5,
- b. Compare N_p and N_b . If $N_p \geq N_b$, do modulus calculation,
$$n = N_p \bmod N_b$$
and, $z = (N_p - n) / N_b$
where, z is an integer value and n is the remainder of the modulus calculation,
- c. $n \in [0, N_b)$, and we skip the first n bits in the Buffer, which is part of the ciphertext that is encrypted with K_1 ,
- d. Decrypt the remaining zN_b bits in the Buffer with K_2 , and replace the ciphertext with the corresponding plaintext,
- e. Decrypt the first N_b bits of the payload data in the Buffer with K_1 , then replace the ciphertext with the corresponding plaintext in the Buffer,

- f. The JPEG image data is re-constructed by combining the original header and the decrypted payload data without any additional bits,

The extreme case is that $n = 0$. Then the first N_b bits of the payload data will
 5 be decrypted with K_2 and K_1 .

$$(7) \quad N_p < N_b$$

For such a case, the ciphertext is merely XORed with the output block from the M Box to get the plaintext, which is shown in the following manner and in Figure
 10 10,

- a. If $N_p < N_b$, an N_b bits block called the Shift Register, which is initially set to some arbitrary value, is used as the input plaintext to the Block Cipher Algorithm,
- b. Use K_1 as the initial Shift Register value, V_r . If the length of the keys, N_k , is less than N_b , just add several dummy bits to the end of K_1 , which
 15 N_k , is less than N_b , just add several dummy bits to the end of K_1 , which may be set to zero, to make V_r the same length as N_b . If N_k is greater than N_b , just take the left-most N_b bits from K_1 as the initial Shift Register value,
- c. Encrypt V_r with K_2 getting the ciphertext of V_c , which is then passed
 20 through the M-Box,
- d. Select the left-most M bits of the ciphertext of V_c , where M is the number of bits in the data block that we wish to decrypt, i.e. N_p ,
- e. The selected M -bit is XORed with the payload data that we wish to decrypt,
- f. The output of the XOR operation is the final plaintext.
 25

In Figure 11 and Figure 12, we show an alternative way for the encryption and decryption of payload data when $N_p \geq N_b$. The details are, as shown in Figure
 11:

- a. Each JPEG image will be loaded to the Buffer and analyzed to
 30 classify the Header and the Payload as defined in Figure 5,

- b. Compare N_p and N_b . If $N_p \geq N_b$, encrypt the last N_b bits of the payload data with K_1 , then replace the plaintext with the corresponding ciphertext in the Buffer,
- c. Do modulus calculation,

$$n = N_p \bmod N_b$$
and, $z = (N_p - n) / N_b$
- where, z is an integer value and n is the remainder of the modulus calculation,
- d. $n \in [0, N_b)$, and we skip the last n bits in the Buffer, which is part of the ciphertext from the above steps that is encrypted with K_1 ,
- e. Encrypt the first zN_b bits in the Buffer with K_2 , and replace the plaintext with the ciphertext,
- f. The JPEG image data is re-constructed by combining the original Header and the encrypted payload data without any additional bits,
- g. The scrambled JPEG image data is fed to the Data Post-processor, if necessary. The process of Data Post-processor will be described later.
- The extreme case is that $n = 0$. Then the last N_b bits of the payload data will be encrypted with K_1 and K_2 . However, as $K_1 \neq K_2$, the last N_b bits are not recovered but encrypted twice.

The corresponding decryption process in Figure 12 is shown as follows:

- a. Each protected JPEG image will be loaded to the Buffer and processed by the Data Pre-processor that will be described later, if necessary. Then it will be analyzed to classify the Header and the Payload as defined in Figure 5,
- b. Compare N_p and N_b . If $N_p \geq N_b$, do modulus calculation,

$$n = N_p \bmod N_b$$
and, $z = (N_p - n) / N_b$
- where, z is an integer value and n is the remainder of the modulus calculation,

- c. $n \in [0, N_b)$, and we skip the last n bits in the Buffer, which is part of the ciphertext that is encrypted with K_1 ,
 - d. Decrypt the first zN_b bits in the Buffer with K_2 , and replace the ciphertext with the corresponding plaintext,
 - 5 e. Decrypt the last N_b bits of the payload data in the Buffer with K_1 , then replace the ciphertext with the corresponding plaintext in the Buffer,
 - f. The JPEG image data is re-constructed by combining the original header and the decrypted payload data without any additional bits,
- 10 The extreme case is that $n = 0$. Then the last N_b bits of the payload data will be decrypted with K_2 and K_1 .

Figure 13 shows another way to realize an EPLE mechanism. The details are:

- 15 a. Each JPEG image is loaded to the Buffer and analyzed to classify the Header and the Payload as defined in Figure 5,
- b. Compare N_p and N_b . If $N_p \geq N_b$, do modulus calculation,

$$n = N_p \bmod N_b$$
and, $z = (N_p - n) / N_b$
- 20 where, z is an integer value and n is the remainder of the modulus calculation,
- c. $n \in [0, N_b)$,
- d. Encrypt the first zN_b bits of the payload data with K_1 , then replace the plaintext with the corresponding ciphertext in the Buffer
- 25 e. Encrypt the last zN_b bits in the Buffer with K_2 , and replace the corresponding data in the Buffer with the ciphertext,
- f. The JPEG image data is re-constructed by combining the original Header and the encrypted payload data without any additional bits,
- 30 g. The scrambled JPEG image data is fed to the Data Post-processor, if necessary. The process of Data Post-processor

will be described later.

The extreme case is that $n = 0$. Then the zN_b bits of the payload data will be encrypted with K_1 and K_2 . However, as $K_1 \neq K_2$, the zN_b bits are not recovered but encrypted twice.

The corresponding decryption process is shown in Figure 14, and the details are shown as follows:

a. Each protected JPEG image is loaded to the Buffer and processed by the Data Pre-processor that will be described later, if necessary. Then it is analyzed to classify the Header and the Payload as defined in Figure 5,

b. Compare N_p and N_b . If $N_p \geq N_b$, do modulus calculation,

$$n = N_p \bmod N_b$$

$$\text{and, } z = (N_p - n) / N_b$$

where, z is an integer value and n is the remainder of the modulus calculation,

c. $n \in [0, N_b)$,

d. Decrypt the last zN_b bits in the Buffer with K_2 , and replace the ciphertext with the corresponding plaintext,

e. Decrypt the first zN_b bits of the payload data in the Buffer with K_1 , then replace the ciphertext with the corresponding plaintext in the Buffer,

f. The JPEG image data is re-constructed by combining the original header and the decrypted payload data without any additional bits,

The extreme case is that $n = 0$. Then the zN_b bits of the payload data will be decrypted with K_2 and K_1 .

In another alternative solution, the encryption sequence of the first zN_b bits and the last zN_b bits of Figure 13 can be exchanged.

In a preferred embodiment of the present invention, if necessary, a Data

Post-processor analyzes the encrypted JPEG image data, to locate Unexpected Marker Codes in the ciphertext, replace the Unexpected Marker Codes with normal data, and generate Unexpected Marker Code Indices. A structure for the Unexpected Marker Code Indices is shown in Figure 15. It can be attached to the end of the JPEG image or saved as an additional file. The embodiment allows a Data Pre-processor to analyze the Unexpected Marker Code Indices and to restore Unexpected Marker Codes before decryption, if Unexpected Marker Codes in the ciphertext have been replaced.

Unexpected Marker Codes are data values generated in the encryption process that accidentally correspond to recognized image file marker codes. Their removal from the ciphertext, can help prevent errors occurring in the scanning of the encrypted files by image handling software and the like, which might otherwise recognize the data values as actual marker codes and act on them accordingly.

Because any Marker Code starts with 0xFF, we only need to replace the prefix 0xFF with any other code, e.g., 0x00 or 0x01, etc. Also, in the Unexpected Marker Code Indices, we only need to record the position of the 0xFF prefixes. The process of Post-processor is shown as follows:

- a. Scan the encrypted payload data to locate any unexpected marker codes,
- b. Replace the prefix (0xFF) of the unexpected marker code with a normal (non-marker code) data code, e.g. 0x01,
- c. Add the location of the replaced prefix to the Unexpected Marker Code Indices, shown in Figure 15, the location being a distance from the first byte of the encrypted payload data.

The process of Pre-processor is shown as follows:

- a. Analyze the Unexpected Marker Code Indices and retrieve the locations of the prefix of the unexpected marker codes,
- b. Restore the prefix (0xFF) of the unexpected marker codes,

In the structure of the Unexpected Marker Code Indices, the start and end

codes are shown as EOI (0xFFD9) marker codes. However, they can be any pair of reserved marker codes in the JPEG standard. Furthermore, the Indices Data should be regarded as COM (comment) data. The length field that follows the COM code is the total size of the Indices Data, which are the locations of the replaced unexpected marker codes.

The replacement of unexpected marker codes in the scrambled payload data may only be necessary when a user wants to decode the encrypted JPEG image forcedly and show mosaics, i.e. images formed from the decoded but still encrypted image data, which will generally take the form of randomly coloured blocks having no recognizable pattern.

It will be understood that although the present invention is particularly useful for JPEG files, and although the above description accordingly explains the invention in relation to JPEGS, the various methodologies and features of the encryption and decryption processes could also be applied to digital image format files in general, where there is a header that contains important information for a decoder to analyze and decode the image data, and a payload that contains the encoded image data. The marker codes of such image files would include for example a header identifier. For example, the methods could be applied to bitmap file format (BMP) images. For example, the bitmap header information could include the header identifier, file size, reserved field, bitmap data offset, bitmap header size, width, height, planes, bits per pixel, compression, bitmap data size, Hresolution, Vresolution, colors, important colors, and palette. The bitmap data following the palette could be regarded as the payload data.

What is claimed is:

1. A method of encrypting an image file of a digital image, the method including the steps of:
 - 5 identifying marker codes in the image file corresponding to file data;
 - identifying payload data in the image file corresponding to image data;
 - applying an encryption process to said payload data to provide encrypted payload data; and
 - constructing an encrypted image file from unencrypted data associated with
 - 10 said marker codes and from said encrypted payload data.
2. The method of claim 1, wherein said encryption process generates encrypted payload data of the same length as said unencrypted payload data.
- 15 3. The method of claim 1 or 2, including the steps of:
 - scanning said encrypted payload data to identify marker codes in said encrypted payload data;
 - replacing identified marker codes with dummy data; and
 - recording replacement data relating to the locations of said replaced marker
 - 20 codes.
4. The method of claim 3, wherein said step of replacing identified marker codes with dummy data includes the step of replacing a common marker code designation by dummy data.
- 25 5. The method of claim 3 or 4, wherein said replacement data is attached to the encrypted image file.
6. The method of claim 3 or 4, wherein said replacement data is recorded in a
- 30 separate file to said encrypted image file.
7. The method of any preceding claim, wherein said digital image is encoded in a JPEG format.
- 35 8. The method of claim 7, wherein said identified marker codes include marker codes in said image file up to and including a first scan header.

9. The method of claim 7, wherein said identified marker codes include SOI, EOI, frame header and first scan header marker codes.
- 5 10. The method of any preceding claim, wherein said encryption process includes a block encryption process, and including the steps of:
identifying a first portion of said payload data having a length corresponding to a multiple of a block size associated with said encryption process; and
10 encrypting said first portion of said payload data using said block encryption process.
11. The method of claim 10, wherein a second portion of said payload data, not forming part of said first portion, is retained as plaintext.
- 15 12. The method of claim 10 or 11, wherein if said payload data is smaller than a block size associated with said encryption process, said payload data is retained as plaintext.
- 20 13. The method of any of claims 1 to 9, wherein a first portion of said payload data is encrypted with a first key, and a second portion of said payload data is encrypted with a second key.
- 25 14. The method of any of claims 1 to 9, including the steps of:
encrypting a first portion of said payload data with a first key to produce a first ciphertext;
combining said first ciphertext with unencrypted payload data to produce mixed payload data;
encrypting a first portion of said mixed payload data with a second key to produce a second ciphertext; and
30 combining said second ciphertext and mixed payload data that has not been encrypted using said second key to produce said encrypted payload data.
- 35 15. The method of claim 14, wherein said first portion of said payload data has a length corresponding to a block size of said encryption process, and wherein said first portion of said mixed payload data has a length corresponding to a multiple of a block size of said encryption process.

16. The method of claim 14, wherein said first portion of said payload data has a length corresponding to a multiple of a block size of said encryption process, and wherein said first portion of said mixed payload has a length corresponding to a block size of said encryption process.
17. The method of claim 14, wherein said first portion of said payload data has a length corresponding to a multiple of a block size of said encryption process, and wherein said first portion of said mixed payload has a length corresponding to a multiple of a block size of said encryption process.
18. The method of claim 15, 16 or 17, wherein said length corresponding to a multiple of a block size is a maximum length of payload data that is a multiple of said block size.
19. The method of any of claims 14 to 18, wherein said first portion of said payload data is taken from the start of said payload data to be encrypted.
20. The method of any of claims 14 to 18, wherein said first portion of said payload data is taken from the end of said payload data to be encrypted.
21. The method of any of claims 14 to 20, wherein when said payload data is smaller than a block size of said encryption process, said payload data is encrypted by the application of a logical operator to the payload data.
22. The method of claim 21, wherein said logical operator is an XOR operator.
23. The method of claim 21 or 22, wherein said payload data is a first operand of said logical operator and a second operand is ciphertext formed by encrypting plaintext using one of said first and second keys.
24. The method of claim 23, wherein said plaintext is based on the other of said first and second keys.
25. A method of decrypting an image file encrypted in accordance with any preceding claim, wherein the method reverses the steps of said claimed encryption

process.

26. A method of decrypting an image file encrypted in accordance with an encryption method of claim 1, including the steps of:

5 identifying plaintext of said encrypted image file corresponding to marker codes of said image file;

identifying ciphertext of said encrypted image file corresponding to payload data;

decrypting said identified ciphertext; and

10 constructing an unencrypted image file from said plaintext and said decrypted ciphertext.

27. A method of decrypting an image file encrypted in accordance with an encryption method of claim 3, including the steps of:

15 identifying marker code replacement data;

forming a modified encrypted image file by replacing dummy data with marker code data in accordance with said identified replacement data; and

decrypting said modified encrypted image file in accordance with claim 26.

20 28. The method of claim 26 or 27, wherein said image file is an image file of a JPEG encoded image.

29. A method of decrypting an image file encrypted in accordance with an encryption method of claim 10, including the steps of:

25 decrypting a first encrypted portion of said payload data having a length corresponding to a multiple of a block size associated with said encryption process to provide plaintext; and

constructing a decrypted image file based on said plaintext and on an unencrypted portion of said image file.

30

30. A method of decrypting an image file encrypted in accordance with an encryption method of claim 14, including the steps of:

identifying a plaintext portion of said image file and a ciphertext portion of said image file;

35 identifying a first portion of said ciphertext portion that has been encrypted by a second key and a second portion of said ciphertext portion that has not been

encrypted by said second key;

decrypted said first portion of said ciphertext using said second key;

forming mixed payload data from said decrypted first portion of ciphertext
and from said second portion of ciphertext;

5 identifying a first portion of said mixed payload data that has been encrypted
by said first key and a second portion of said mixed payload data that has not been
encrypted by said first key;

decrypted said first portion of said mixed payload data using said first key;
and

10 forming a decrypted image file from said plaintext portion of said image file,
from said decrypted first portion of said mixed payload data and from said second
portion of said mixed payload data.

31. An encryption system for encrypting an image file of a digital image, the
15 system including:

means for storing an image file;

means for scanning said image file to identify marker codes within said file
corresponding to file data and to identify payload data within said file corresponding
to image data;

20 means for applying an encryption process to said payload data to provide
encrypted payload data;

means for constructing an encrypted image file from said encrypted payload
data and from unencrypted data associated with said marker codes.

25 32. The system of claim 31, wherein said encryption process generates
encrypted payload data of the same length as said unencrypted payload data.

33. The system of claim 31 or 32, including:

30 means for scanning said encrypted payload data to identify marker codes in
said encrypted payload data;

means for replacing identified marker codes with dummy data; and

means for recording replacement data relating to the locations of said
replaced marker codes.

35 34. A decryption system for decrypting an image file of a digital image
encrypted by an encryption method of claim 1, the system including:

means for storing an encrypted image file;
means for identifying plaintext of said encrypted image file corresponding to
marker codes of said image file;
means for identifying ciphertext of said encrypted image file corresponding
5 to payload data;
means for decrypting said identified ciphertext; and
means for constructing an unencrypted image file from said plaintext and
from said decrypted ciphertext.

10 35. A decryption system for decrypting an image file of a digital image
encrypted by an encryption method of claim 3, including:

means for storing an encrypted image file;
means for identifying marker code replacement data in said encrypted image
file;
15 means for forming a modified encrypted image file by replacing dummy
data with marker code data in accordance with said replacement data; and
means for identifying plaintext of said modified encrypted image file
corresponding to marker codes of said image file;
means for identifying ciphertext of said modified encrypted image file
20 corresponding to payload data;
means for decrypting said identified ciphertext; and
means for constructing an unencrypted image file from said plaintext and
from said decrypted ciphertext.

25 36. A computer-readable data carrier having program code recorded therein,
said program code, when run on a computer, implementing an encryption or
decryption process according to any of claims 1 to 30.

30 37. An encrypted image file, said encrypted image file including ciphertext
corresponding to encrypted image data of a digital image and plaintext corresponding
to image file marker codes and marker code data, said plaintext and ciphertext being
combined according to the data structure of an image file, such that a processing
engine for said image files is able to extract information regarding said image file
and/or said digital image from said encrypted image file without decryption of said
35 encrypted image file.

38. The encrypted image file of claim 37, wherein said ciphertext is of the same length as the plaintext image data that has been encrypted in the encrypted image file.

39. The encrypted image file of claim 37 or 38, including marker code
5 replacement information indicating locations in said ciphertext where data is to be replaced by marker code data prior to decryption.