ELSEVIER Signal Processing: *Image Communication* 10 (1997) 279–290

SIGNAL PROCESSING: IMAGE COMMUNICATION

A linear quadtree compression scheme for image encryption[1]

Henry Ker-Chang Chang*, Jiang-Long Liu

Graduate School of Resources Management, National Defense Management College, P.O. Box 90046-15, Chung-Ho, Taipei, Taiwan, ROC

Received 9 January 1995

Abstract

A private key encryption scheme for a two-dimensional image data is proposed in this work. This scheme is designed on the basis of lossless data compression principle. The proposed scheme is developed to have both data encryption and compression performed simultaneously. For the lossless data compression effect, the quadtree data structure is used to represent the image; for the encryption purpose, various scanning sequences of image data are provided. The scanning sequences comprise a private key for encryption. Twenty four possible combinations of scanning sequences are defined for accessing four quadrants, thereby making available $24^n \times 4^{n(n-1)/2}$ possibilities to encode an image of resolution $2^n \times 2^n$. The security of the proposed encryption scheme therefore relies on the computational infeasibility of an exhaustive search approach. Three images of 512×512 pixels are used to verify the feasibility of the proposed scheme. The testing results and analysis demonstrate the characteristics of the proposed scheme. This scheme can be applied for problems of data storage or transmission in a public network. © 1997 Elsevier Science B.V.

Keywords: Linear quadtree: Image compression; Image encryption; Private key; Image scramble

1. Introduction

Important images must be protected prior to their transmission to the recipients, and encryption is an effective method for this purpose. A two-dimensional (2-D) image is generally raster-scanned. The output of the scanning can be viewed as a list of one-dimensional (1-D) image data with which image pixels are encrypted and transmitted. Various encryption schemes can therefore be applied to protect 1-D image data after the 2-D image has been scanned. Among various data encryption schemes, both block cipher [5] and stream cipher [1, 7] are appropriate for image encryption. A block cipher, DES [2, 4] as an example, divides the message (plaintext) M into successive x blocks $M_1, M_2, \dots, M_x$ and enciphers each block M_i with a specific encryption key k, i.e., $E_k(M) = E_k(M_1) E_k(M_2) \dots E_k(M_x)$. Stream cipher is a different approach which breaks the original message M into successive s bits $b_1, b_2, \dots, b_s$ and enciphers each b_i with the ith element k_i of a key stream $K = k_1k_2 \dots k_s$; i.e., $E(M) = E_{k1}(m_1)E_{k2}(m_2) \dots E_{ks}(m_s)$. Although both methods have security over data

*Corresponding author. E-mail: chang@rs360.ndmc.edu.tw; tel.: (886)-2-2222137 ext. 8563; fax: (886)-2-2232876.

[1] Part of this paper was presented in 1994 International Computer Symposium, Hsinchu, Taiwan, ROC.

channel, a significant amount of time spent in scanning and enciphering is inevitable. Additionally, reducing the transmission cost as well as the number of data transmitted would require an extra amount of time to compress the original data prior to transmission. Bourbakist and Alexopoulos [3] have developed another image encryption technique. Their approach has an image encrypted by the SCAN language. The SCAN language is a simple context-free language devoted to accurately describing and generating a wide range of 2-D array accessing algorithms. They define several SCAN patterns, naming SCAN alphabets, as basic elements for image accessing sequences. Each SCAN alphabet represents one kind of scan order. Different kinds of combinations of SCAN alphabets, referred to as a SCAN word, may produce different kinds of secret image data. Their method is also secure without any effect of data compression.

Image scramble [8] is generally applied to the 2-D image whenever it is desirable to simplify image processing to transmission or storage. Scramble method, being used popularly in TV signal encryption, views the original image as a 2-D array and encrypts the 2-D data by means of transposition cipher. It encrypts an image by transposing certain areas of the image line by line. Therefore, hardly any arithmetic operation is required in encryption. This kind of method has the advantage of fast image encryption; however, it is not secure enough to encrypt important image data. Besides, it still must waste comparison time prior to transmission.

According to the above-mentioned description, neither the encryption nor the scrambling can obviously satisfy both objectives of data compression and encryption simultaneously. Moreover, the encrypted and compressed image data mean nothing to a recipient except for the fact that the recipient must decompress and decrypt the received data before reading it. For the purposes of security and shortening storage space, an important image must be encrypted and compressed before it is either stored in a secondary device or transmitted. In this paper, a new image encryption scheme is proposed with which the problem of data compression is also solved, i.e., a technique combining both image encryption and compression together. The proposed scheme is developed on the basis of quadtree data structure and secret scanning orders. The proposed technique, capable of encrypting image data at the same time the image is scanned, has the advantage of fast processing. Consequently, transmission time is saved as a result of the efficiency of data compression. Moreover, the codes after encryption may be processed directly for it is constructed on the basis of the quadtree data structure. This direct processing capability implies that any user may manipulate the encrypted image data without using any one of the following routines of encryption–compression, decompression–decryption or reencryption–redecompression.

The rest of this paper is organized as follows. In Section 2, the proposed approach for both purposes of image compression and encryption is described. Image compression is first introduced; the method for encryption is then derived; the protocol and the detailed algorithm of the proposed scheme is finally explained. In Section 3, several images are tested to illustrate the feasibility of the proposed scheme. In Section 4, the security of the proposed scheme is analyzed. Concluding remarks are provided in the last section.

2. The proposed image encryption with lossless compression

The proposed image encryption scheme based on the principle of lossless data compression is provided in this section. The motivation behind this scheme is to have the image encrypted and compressed simultaneously. Thus, the proposed scheme includes two characteristics, i.e., the security of the image data can be ensured and the quantity of data can be condensed.

2.1. Image compression

For a $2^n \times 2^n$ image, n being the image resolution, the image can be recursively subdivided into four equal-sized quadrants until all the image elements (i.e., pixels) within a quadrant have the same gray value. Thus, the hierarchical representation of a $2^n \times 2^n$ image can be derived and is considered as a quadtree data structure. Each nonleaf node in the quadtree contains mixed gray values and has four sons. The relationship between a father node and its children nodes is embedded in the corresponding linear quaternary codes. With the quadtree representation, the root node labeled level n corresponds to the entire image and its each son node at level $n - 1$ contains one-fourth the image size. All leaf nodes of the tree correspond to those blocks in which uniform intensity is found and no further subdivision is necessary. Therefore, a fully complete quadtree has root node at level n while a node at level 0 corresponds to an image pixel in the image. For clear illustration of a quadtree data structure, the sample image shown in Fig. 1(a) functions as an illustrative example. The image is a $2^3 \times 2^3$ binary image (i.e., an image contains only two kinds of color). Note that all 1's in the sample image correspond to pixels which have white color, and all 0's correspond to pixels which have black color. The decomposed image for the sample image is shown in Fig. 1(b); in addition, the final quadtree representation for Fig. 1(b) is shown in Fig. 1(c). The levels of the tree are also marked. In Fig. 1(c), each pixel is numbered in Morton sequence [8]. The Morton sequence provides a relatively easy transformation between the coordinates of a pixel in the image and the Morton number. With the usage of Morton numbers, the position of a pixel in the image can be easily derived.

The effect of image compression can then be found as the original image is encoded using quadtree data structure. This is because only the leaf nodes of the quadtree are recorded. If the length of the bit string required for the representation of leaf nodes is less than the original number of pixels, the compression result is definite. In the proposed scheme, an image is encoded by recording only the leaf nodes, i.e., each leaf node is encoded by a bit string consisting of a gray value and a level indicator. The level indicator l has duplicate l marks if the leaf node is on level l. The mark used here is x. Taking Fig. 1(a) as an example again, the final code string is derived as '0x00100x1x0xx1xx10100x1x0x' in which two bits are used to encode '0', '1' and the mark 'x', so 52 bits are used. It is less than the original code string of 64 bits. Notably, the mark

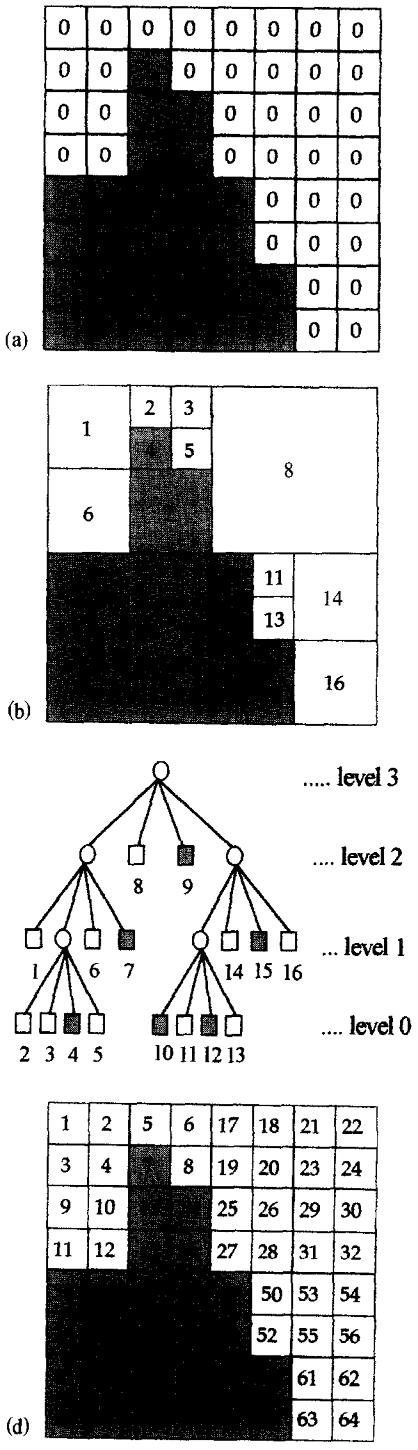


Fig. 1. (a) A sample binary image. (b) The decomposed image. (c) The quadtree representation. (d) The Morton sequence of the sample image.

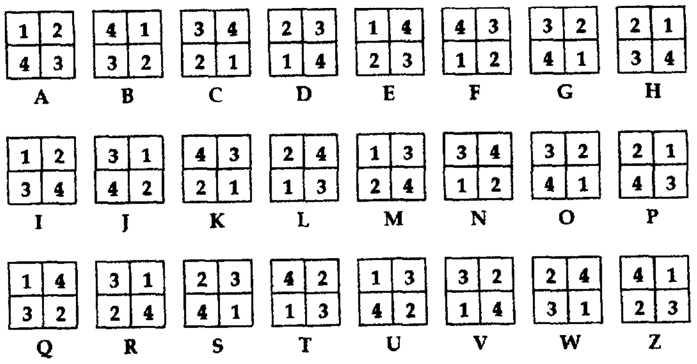


Fig. 2. 24 scan units for image encryption.

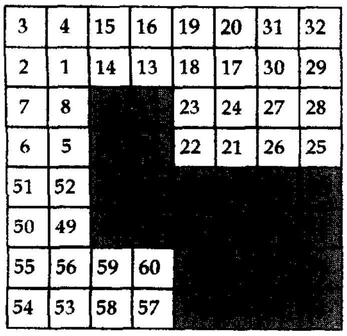
can be of any form which is distinguishable from the gray value.

The compression capability of the proposed scheme evolves from the coding result of the quadtree. The encryption method for secrecy preserving is next described. From the above-mentioned description, a quadtree is apparently constructed using the top-down process. In practice, however, a quadtree for an input image is constructed by bottom-up process. The image is scanned following the Morton sequence in order to construct a quadtree. The Morton sequence for the sample image is shown in Fig. 1(d). The scanning sequence for the image is similar to scanning each of the four quadrants following the writing sequence of letter Z. For example, to construct a quadtree in Fig. 1(c), Fig. 1(a) is scanned following the number sequence in Fig. 1(d). The encryption of image data is therefore inspired to scan the image in various scanning sequences and have the scanning sequences not disclosed.

2.2. Image encryption

The basis of the proposed image encryption scheme is the secrecy of the scanning sequences for the image, i.e., each internal node of the quadtree is scanned by a scanning sequence different from all others. As generally known, 24 possible scan sequences can be found when an internal node is partitioned into four quadrants; it implies that 24^n possible sequences are available to scan an area at the first level in a quadtree. The degree of security is then derived on the basis of secrecy of scanning sequences. For an image of resolution $2^n \times 2^n$, a quadtree with n levels is derived. Because the quadtree for the image is built in bottom-up process, it implies that all the nonleaf nodes in the complete quadtree have to be scanned. If each nonleaf node is scanned by a different order, then there are $4^n \times 2^{n(n-1)/2}$ (the derivation is stated later) combinations to encode the whole image. If the scanning sequence is not disclosed, the only possibility of breaking the encrypted codes is to guess one out of $4^n \times 2^{n(n-1)/2}$ combinations. A brute force approach involves finding the right one by attempting all $4^n \times 2^{n(n-1)/2}$ possibilities, which is generally accepted to be impossible. In order to describe the proposed encryption scheme, 24 scan units are defined as shown in Fig. 2. Each scan unit is represented by an alphabet (termed as scan letter); i.e., each scan letter represents a specific sequence to scan four quadrants for a nonleaf node in the quadtree.

Several scan letters comprise a scan word which provides the access sequence to nonleaf nodes in a quadtree for a $2^n \times 2^n$ image. The scan word has the form $W = W_n W_{n-1} \ldots W_1$, in which W_i denotes the scan sequence for nodes in level i; each W_i has the form $W_i = L_1 L_2 \ldots L_t$, $t = 4^{n-i}$, in which each L denotes a scan letter applied to scan four quadrants for a node. In the proposed encryption scheme, the scan word is considered as an encryption key. We may think of a scan function which is used to generate all scan sequences for the image. The scan function is intended to be able to produce

(a)


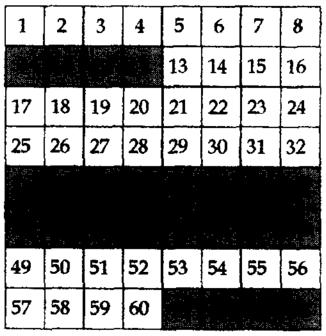
(b)


Fig. 3. (a) The scan sequence of a sample image. (b) A restored image using run length coding scheme.

unpredictable sequences of scan letters. A sample image of size $2^3 \times 2^3$ shown in Fig. 3(a) is used to illustrate the function of the scan word. For simple explanation, we assume that all nodes on the same level are scanned by the same scan sequence and the scan sequences are selected from the scan letters as A, E and C for levels 3, 2 and 1, respectively. Thus, $W_3 = \text{A}$, $W_2 = \text{EEEE}$ and $W_1 = \text{CCCCCCCCCCCCCCCC}$. Therefore, the scan word $W = W_3W_2W_1$ is applied to scan the image. The root node is scanned by a sequence represented by letter A; the nodes in level 2 are scanned by letter E sequence and the third level has all nodes scanned by letter C. The access sequence is shown in Fig. 3(a) in which each pixel is numbered according to the scanning sequences of letters C, E and A. Fig. 3(b) depicts an incorrect image decrypted using a way similar to the run length coding scheme. Fig. 3 clearly indicates that the encrypted image is secure if the scan word is not disclosed. Besides, the encoded bit is '0x0x1x0x0xx1xx0x0x0x1x' which costs only 44 bits, i.e., a reduction of 34% is found corresponding to the original image of 64 bits.

2.3. *Encryption/decryption protocol and algorithms*

How the proposed scheme combines the effect of encryption with compression has so far been described. In this section, the protocol and the algorithm for the encryption/decryption are described.

2.3.1. *Protocol*

Since the proposed encryption/decryption scheme is symmetric, the following protocol is applied:
Step 1. Sender generates the scan fuction F and a seed number s.
Step 2. Sender sends (F, s) to the receiver via a secure channel.
Step 3. Sender encrypts the image data with the scan function F by $C = E_F(M)$, where C denotes ciphertext, M denotes source image data.
Step 4. Sender sends the ciphertext C to receiver through an insecure channel.
Step 5. Receiver recovers the source image M with the scan function F and seed number s by $M = D_F(C)$.

2.3.2. *Encryption/decryption algorithm*

The encryption algorithm consists of two procedures. The first procedure involves finding where the location is for the next pixel to be processed? The second procedure involves generating the corresponding quaternary codes.

In the first procedure, the image is scanned with the traversal order of depth first search (DFS). This task involves determining the position of the next pixel or block. The position of the pixel to be processed depends on the scan word for encryption. According to the predefined scan word, the position of each pixel is determined by the sum of the displacements in each level. Notably, the term 'pixel' here refers to a leaf node while the term 'block' represents a nonleaf node. Pixel and block are alternatively used as the encryption procedure is described. To find the position of the next pixel (block) b in level l, let $\Delta X_{Li}(l, b)$ denote the displacement in x-direction for the next block b by scan letter Li in level l; $\Delta Y_{Li}(l, b)$ denote the displacement in y-direction for the block b by scan letter Li in level l. Consequently, the displacements in x- and y-directions can be determined for these 24 scan units according to various situations as shown in the following:

$\Delta X_A = \text{EQ1}, \ \Delta Y_A = \text{EQ4};$

$\Delta X_B = \text{EQ2}, \ \Delta Y_B = \text{EQ1};$

$\Delta X_C = \text{EQ3}, \ \Delta Y_C = \text{EQ2};$

$\Delta X_D = \text{EQ4}, \ \Delta Y_D = \text{EQ3};$

$\Delta X_E = \text{EQ4}, \ \Delta Y_E = \text{EQ1};$

$\Delta X_F = \text{EQ1}, \ \Delta Y_F = \text{EQ2};$

$\Delta X_G = \text{EQ2}, \ \Delta Y_G = \text{EQ3};$

$\Delta X_H = \text{EQ3}, \ \Delta Y_H = \text{EQ4};$

$\Delta X_I = \text{EQ5}, \ \Delta Y_I = \text{EQ4};$

$\Delta X_J = \text{EQ2}, \ \Delta Y_J = \text{EQ5};$

$\Delta X_K = \text{EQ6}, \ \Delta Y_K = \text{EQ2};$

$\Delta X_L = \text{EQ4}, \ \Delta Y_L = \text{EQ6};$

$\Delta X_M = \text{EQ4}, \ \Delta Y_M = \text{EQ5};$

$\Delta X_N = \text{EQ5}, \ \Delta Y_N = \text{EQ2};$

$\Delta X_O = \text{EQ2}, \ \Delta Y_O = \text{EQ6};$

$\Delta X_P = \text{EQ6}, \ \Delta Y_P = \text{EQ4};$

$\Delta X_Q = \text{EQ5}, \ \Delta Y_Q = \text{EQ1};$

$\Delta X_R = \text{EQ3}, \ \Delta Y_R = \text{EQ1};$

$\Delta X_S = \text{EQ6}, \ \Delta Y_S = \text{EQ3};$

$\Delta X_T = \text{EQ1}, \ \Delta Y_T = \text{EQ6};$

$\Delta X_U = \text{EQ1}, \ \Delta Y_U = \text{EQ5};$

$\Delta X_V = \text{EQ5}, \ \Delta Y_V = \text{EQ3};$

$\Delta X_W = \text{EQ3}, \ \Delta Y_W = \text{EQ6};$

$\Delta X_Z = \text{EQ6}, \ \Delta Y_Z = \text{EQ1}.$

Here each ΔX_{Li} is abbreviation of $\Delta X_{Li}(l, b)$ and

$\text{EQ1} = (-1)^{b \bmod 3}[(b+1) \bmod 2](2^n/2^{n-l}),$

$\text{EQ2} = (-1)^{b+1 \bmod 3}(b \bmod 2)(2^n/2^{n-l}),$

$\text{EQ3} = (-1)^{b-1 \bmod 3}(b \bmod 3)(2^n/2^{n-l}),$

$\text{EQ4} = (b \bmod 2)[(b-1)/2](2^n/2^{n-l}),$

$\text{EQ5} = (-1)^b[(b+2)/4](2^n/2^{n-l}),$

$\text{EQ6} = (-1)^{b+1}(2^n/2^{n-l}).$

The above-mentioned six equations are derived by fully considering all the variations for these 24 scan units. Thus, for a $2^n \times 2^n$ image, the position P of a pixel can be determined by the sum of the displacements of each level:

$$P = 2^n \Delta Y + \Delta X,$$

where

$$\Delta X = \sum_{i=1}^{n} \Delta X_{L_i}(i, b_i), \quad \Delta Y = \sum_{i=1}^{n} \Delta Y_{L_i}(i, b_i).$$

The first procedure is more clearly explained by using the sample image in Fig. 3(a) to find the position of the first pixel. Since the scan word is AEC, the displacements of each level are computed by $\Delta X_A = \text{EQ1}$, $\Delta X_E = \text{EQ4}$ and $\Delta X_C = \text{EQ3}$. For all equations, b equals 1. Therefore, a displacement of 1 for the x-direction is derived by the summation of zero in level 1, zero in level 2 and one in level 3. In the same manner, displacement ΔY is also 1. The position of the first pixel to be processed should clearly be (1, 1) on the image. As the position of a pixel is found, the quaternary code for this pixel is determined according to the quadtree coding method. While scanning the image, the intensity value and level number of a pixel (block) is recorded if four quadrants contain only the same intensity value; in addition, all the pixels are output if the four quadrants are of mixed intensity values. The encryption algorithm is listed in Appendix A.

The decryption algorithm is simple. First, the compressed codes are expanded in reverse order. Next, exact positions of pixels are computed according to the scan word and the level marks. The intensity value of a pixel is then derived according to the intensity value in the encoded bit string. Therefore, the entire image can be completely restored without any information loss. The decryption algorithm is listed in Appendix B.

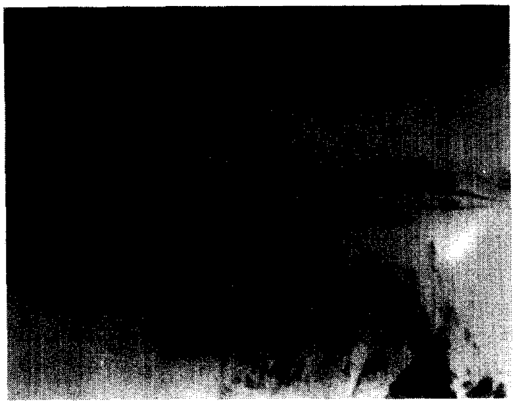
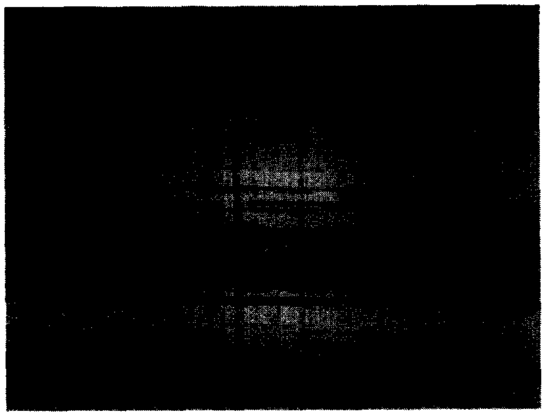
(a)

(b)

(c)

Fig. 4. (a) F-16 fighter. (b) A restored image using encrypted data. (c) A restored image using run length coding scheme.

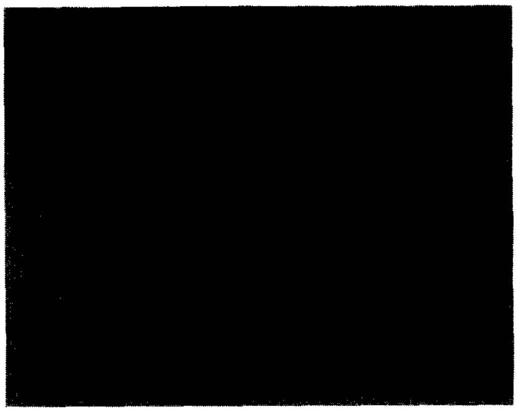
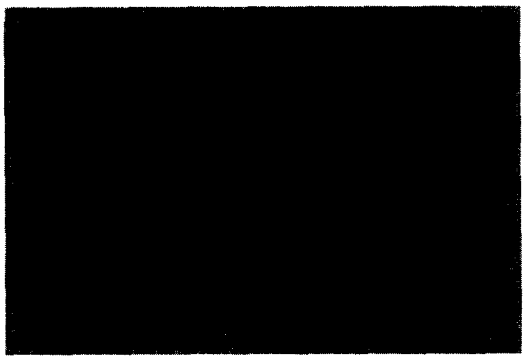
(a)

(b)

(c)

Fig. 5. (a) A floppy diskette. (b) A restored image using encrypted data. (c) A restored image using runlength coding scheme.

3. Empirical tests and security analyses

The feasibility and advantages of the proposed scheme are discussed in this section. The degree of security for the proposed scheme is also analyzed. Several empirical tests are implemented. A comparison of the results is made to demonstrate the characteristics of the proposed scheme.

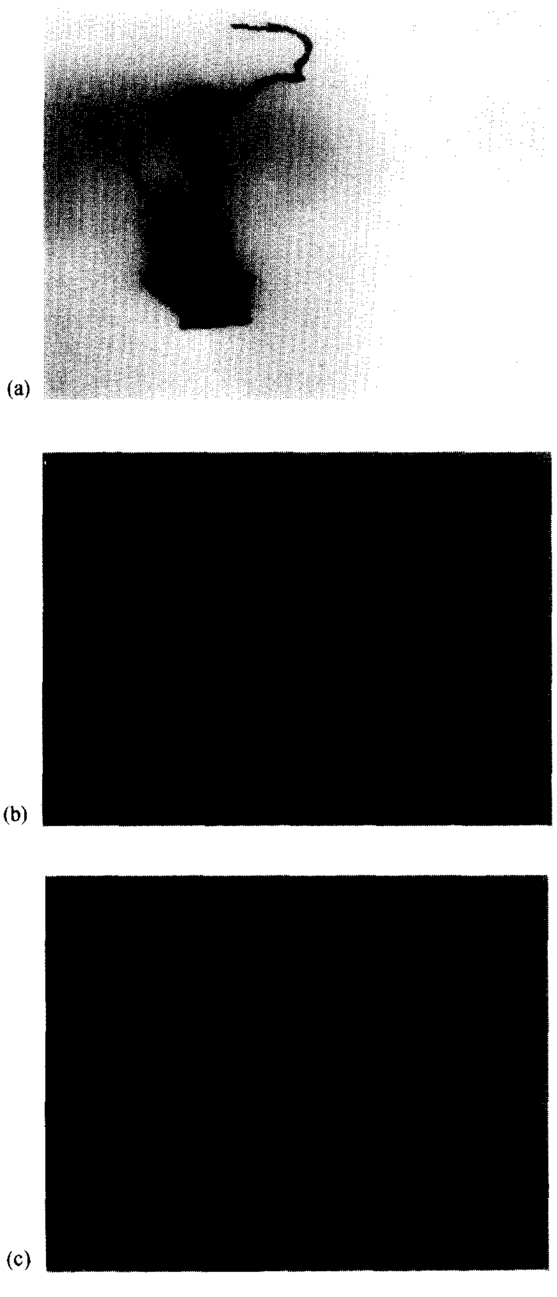

(a)
(b)
(c)

Fig. 6. (a) A regional data image. (b) A restored image using encrypted data. (c) A restored image using runlength coding scheme.

3.1. Empirical tests

The proposed encryption and decryption algorithms are implemented using C language in which two images of resolution $2^9 \times 2^9$ with 256 gray levels and a binary image are tested. The first image is an F-16 fighter as shown in Fig. 4(a), the second image is a floppy diskette shown in Fig. 5(a) and the third binary image of a regional data is shown in Fig. 6(a). In the experimental tests, the scan letter is selected as $W = W_9 W_8 \cdots W_1$. *Again, for a simple explanation, we assume that all nodes on the level are scanned by the same scan sequence and the scan sequences are recursively selected from the scan function as* A, E *and* C *through levels* 9 *to level* 1. The feasibility of the proposed scheme is demonstrated by restoring the images using the encoded quaternary code first. Next, the run length coding scheme [9] is also used to restore images. The run length coding scheme is used here because not only it is popular for image compression, but also the encoded quaternary codes are quite similar to the output of the run length codes. The tests are used to reflect the impossibility of restoring images if the scan word is not known. Figs. 4(b), 5(b) and 6(b) are the images restored from the quaternary codes after encryption using the scan word W. From these restored images, the first primary objective of image encryption is achieved. The encrypted images contain nothing for the viewer. Additionally, the images would not work if the illegal user intends to restore the original image by the run length coding scheme. The images shown in Figs. 4(c), 5(c) and 6(c) are the results derived by the run length decoding scheme. These figures indicate that the original images cannot be found if the encrypted key is not disclosed.

Concerning the effect of data compression, the F-16 image of size 262 144 bytes is diminished to 259 866 bytes, a shortened size of 3278 bytes is gained; to the image of floppy diskette, the original 262 144 bytes are decreased to 212 886 bytes, almost of 20% of the original image size is shortened; to the image of regional data (binary image), the original 32 768 bytes are decreased to 11 678 bits, almost 95% are decreased. All statistics of empirical tests for the compression effect are collected in Table 1. The second objective of the proposed scheme for data compression is completely reached.

Empirical results indicate that the encrypted images cannot be recognized anyway and also the compression effects can be affirmed.

Table 1
The compression effect of empirical tests

Gray image	Plaintext	Ciphertext	Compression
F-16 Fighter	262 144 bytes	259 866 bytes	1%
Diskette	262 144 bytes	212 886 bytes	20%
Regional data	3278 bytes	11 678 bytes	95%

Combining the motivation of the proposed scheme with the results of the empirical tests reveals two noteworthy points. First, the proposed image encryption scheme is secure enough under the condition that the scan word is safely kept. That is, the encryption scheme essentially belongs to the conventional private key cryptosystem. The reason why the private key cryptosystem is applied here is due to the relatively easy computation of the encryption/decryption processes. Although the public key cryptosystem seems to be more attractive than the private key cryptosystem, the heavy computational burden and the degree of reliability is not yet acceptable. On the contrary, the private key cryptosystem is still in use. Characteristics of light computation and easy implementation are the major factors which are desirable. Second, the compression effect of the quadtree data structure depends on the complexity of the image. The more complex the image is the longer the encoded bit string will be. The outputs of the empirical tests verify this fact. The reason is that there are too many leaf nodes on the lower levels for the image with complex data. A complex image implies that it is less possible to compact during the construction of a quadtree. From the empirical tests, the floppy diskette is obviously more simple than the F-16 fighter, thereby the compression result is sharply improved. However, the compression effect would always be better than storing the original image since any compacting can induce a compression effect.

3.2. Security analysis

How is the security of the proposed encryption scheme? This is a key point to the feasibility of the proposed encryption approach. Since the security of the proposed image encryption scheme depends on the privacy of the encryption key, the only attack we have to consider is whether or not the proposed encryption scheme is immune to the brute force searches; in other words, can the compromise of the proposed encryption scheme be computationally infeasible? The degree of security for the proposed scheme relies on how many combinations of the scan word there can be. Given an image of $2^n \times 2^n$, a quadtree of height n is built. According to the encryption scheme described above, there are 24 combinations to encode a partitioned region, so there are 24 combinations for the first partition; at the second partition, each of the four regions has 24 scan combinations, 4×24 combinations are derived; at the third partition, 16×24 combinations are found; by the way, the derivation of combinations for each level continues until $4^{n-1} \times 24$ combinations are found for the last run, the $n-1$ partition. Therefore, total combinations for the generation of scan word, TC, can be

$$\begin{aligned} \mathrm{TC} &= 24 \times (4 \times 24) \times (16 \times 24) \times \cdots \times (4^{n-1} \times 24) \\ &= 24^n \times (4^0 \times 4^1 \times 4^2 \times \cdots \times 4^{n-1}) \\ &= 24^n \prod_{i=0}^{n-1} 4^i \\ &= 24^n \times 4^{n(n-1)/2}. \end{aligned}$$

For $n = 9$ as an example, there are about 249 272 combinations to generate all scan words. If an illegal user applies a computer of 100 MIPS to break the proposed encryption scheme, a

Table 2
Various computational loads for different data n

n	$24^n \times 4^{n(n-1)/2}$	100 MIPS processor
1	24	2.4E-7 s
2	2.30E3	2.3E-5 s
3	8.85E4	8.85E-4 s
4	1.36E9	13.6 s
5	8.35E12	23.19 h
6	2.05E17	65 yr
7	2.02E22	6.41E7 yr
8	7.93E27	2.52E13 yr
9	1.25E34	3.96E29 yr

computational load of

$$\frac{24^9 \times 2^{72}}{100 \times 10^6 \times 3600 \times 24 \times 365},$$

which is about 3.96×10^{18} years is required to exhaustively search a scan word. Table 2 lists various computational loads for different data *n*. This implies the rigidity of the proposed image encryption scheme.

4. Conclusion

A scheme combining encryption and compression for 2-D image data is proposed in this study. This proposed scheme is developed to encrypt and compress the image data simultaneously. The encryption method is designed on the basis of lossless data compression. While the scanning word, encryption key, is not disclosed, various combinations of scanning sequences for nodes at various levels in a quadtree would cause the image to be scrambled randomly. Therefore, the image will be secure no matter whether the image is stored in the secondary storage or transmitted in networks. A compression effect is also provided in the proposed scheme. The quadtree data structure is used to represent an image. The result of data compression appears when any data compaction is found as the quadtree is constructed. Results obtained from the empirical tests and analyses verify the proposed scheme to be applicable to image transmission or storage. Future research efforts could emphasize:

1. applying a public key cryptosystem for encryption, and
2. finding a more efficient lossless data compression scheme.

Appendix A. Encryption algorithm

```
procedure ENCRYPT(s, F, SOURCE, TARGET, N)
/* This procedure entails encrypting a $2^N \times 2^N$ image data which are stored in array SOURCE
and the encoded words are stored in array TARGET. */
input:
F/* the scan function used to randomly generate a number ranged from 1 to 24.*/
s/* the seed number of F*/
SOURCE /* the image data going to be encrypted */
N/* size of source image */
output:
TARGET /* the encrypted data */
procedure NEXE_BLOCK(clevel)
*/ This procedure recursively performs the DFS */
begin
if clevel = flevel then
for i: = 0 to 3 do /* scan the four leafnodes */
T(i): = SOURCE(NEXT(F)) /* NEXT(F) is used to determine the position of next pixel */
end for;
if T(i) are equal /* the four leafnodes have the same value */
plevel: = clevel-1
$T_{plevel}(B_{plevel})$.flag: = 1
$T_{plevel}(B_{plevel})$.value: = T(0)
else /* output the four leafnodes */
for i: = 0 to 3 do
TARGET(P): = T(i);
```

```
P:= P + 1;
end for;
end else;
else
if $B_{clevel} < 4$ then
nlevel:= clevel + 1;
NEXT_BLOCK(nlevel);
else
if $T_{clevel}$(i) are equal /* record the status of current block */
plevel:= clevel-1;
$T_{plevel}(B_{plevel})$.flag:= 1;
$T_{plevel}(B_{plevel})$.value:= $T_{clevel}$(0).value;
else /* output all the recorded pixels and marks*/
for j:= 0 to 3 do
if $T_i$(j).flag = 1 then
TARGET(P):= $T_i$(j).value;
$T_i$(j).flag:= 0;
P:= P + 1;
end if;
for k:= 0 to flevel-plevel
TARGET(P):= 'x';
end for;
end for;
end if;
end if;
end if;
end NEXT_BLOCK;
begin
flevel = N; /* final level */
x = Find_x(SOURCE); /* find a gray level used to represent a separator and derives the
marks in a linear quaternary code. */
clevel:= 0;
NEXT_BLOCK(clevel);
end ENCRYPT;
```

Appendix B. Decryption algorithm

```
procedure DECRYPT(s, F, SOURCE, TARGET, TEMP, N, x)
/* This procedure decrypting a $2^N \times 2^N$ image data which are stored in array SOURCE and the
decoded words are stored in array TARGET.*/
input:
F /* the scan function used to randomly generate a number ranged from 1 to 24. */
s /* the seed number of F */
SOURCE /* the image data going to be encrypted */
N /* size of source image */
x /* the separator */
```

```
output:
TARGET /* the decrypted data */
TEMP /* the result array of run-length coding */
begin
RUN_LENGTH(x, SOURCE, TEMP) /* using run-length coding to expand the SOURCE */
TAGET(i) = NEXT(TEMP, F); /* find the next pixel in the TEMP and output to TAGET */
end DNCRYPT;
```

References

[1] H. Beker and F. Piper, *Cipher Systems: The Protection of Communications*, Wiley, New York, 1982.

[2] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems", *Proc. Crypto 90*, Santa Barbara, CA, August 1990, p. 17.

[3] N. Bourbakist and C. Alexopoulos, "Picture data encryption using scan patterns", *Pattern Recognition*, Vol. 25, No. 6, 1992, pp. 567–581.

[4] R.L. Davis, "Some rebular properties of the data encryption standard", in: *Advances in Cryptology-Proc. Crypto 82*, Plenum Press, New York, 1982, pp. 89–96.

[5] H. Feistel, Block cipher cryptographic system, US Patent No. 3, 798, 359, March 1974.

[6] G.M. Morton, *A Computer Oriented Geodetic Data Base, and A New Technique in File Sequencing*, IBM Canada Ltd.

[7] R.A. Rueppel, *Analysis and Design of Stream Ciphers*, Springer, Berlin, 1986.

[8] J. Slater, "Scramble (TV signal encryption)", *Electron. Today Int. (UK)*, Vol. 19, No. 2, 1990, pp. 16–20.

[9] R.N. Williams, *Adaptive Data Compression*, Kluwer Academic Publishers, Boston, 1991.