On The Application of Image Decomposition to Image Compression and Encryption

Howard Cheng and Xiaobo Li
University of Alberta
Department of Computing Science, GSB 615, University of Alberta,
Edmonton, Alberta, Canada, T6G 2H1.
Phone: (403) 492-2299, Fax: (403) 492-1071
email: hcheng@cs.ualberta.ca, li@cs.ualberta.ca

Abstract

The recent explosion in multimedia and networking applications places a great demand on efficient data storage and transmission techniques. In addition, network communication, especially wireless communication, can easily be intercepted. Therefore, sensitive information must be protected from eavesdroppers. These two problems have traditionally been treated independently of each other. In this paper, we will study a new image compression and encryption scheme that allows any conventional encryption algorithm to take advantage of the properties of a class of image compression algorithms based on image decomposition. An image compression algorithm that produces several different parts is used. The encryption process identifies the parts that are crucial and encrypts only those parts. The security of this scheme and some possible applications will be discussed in this paper.

Keywords

Image decomposition, quadtrees, image compression, encryption

1 INTRODUCTION

As multimedia applications become more popular, there is a greater demand for the efficient representation of many different types of data. To accommodate the increasing use of multimedia in network environments, the amount of data transmitted must be minimized. Since network communication, especially wireless communication, can be intercepted, sensitive information must be protected from unauthorized access. Images are used heavily in multimedia applications, so a good compression and encryption scheme for images will have many applications. In this paper,

This research is supported in part by the Motorola Wireless Data Group and the Canadian Natural Sciences and Engineering Research Council under Grant OGP9198.

we discuss a general scheme in which image decomposition can be applied to image compression and encryption. This scheme is aimed at wireless applications in which portable units may have limited computing resources.

Traditionally, compression and encryption algorithms have been developed independently of each other. The compressed data is encrypted with an encryption algorithm that does not take advantage of the properties of the chosen compression algorithm. An alternative scheme is discussed in this paper. In this new approach, an image compression algorithm that produces a number of different parts is used. Some of these parts are crucial from the encryption point of view while others are not. Thus, only the crucial parts need to be encrypted.

This paper is organized as follows. First, an overview of our scheme and some possible applications are presented in Section 2. Next, in Section 3, the security of our scheme using frequency domain decomposition is examined. We described a way to apply our scheme to spatial domain decomposition in Section 4, and finally, its security is discussed in Section 5.

2 OVERVIEW AND POSSIBLE APPLICATIONS

In this section, we present an overview of our scheme and some of its possible applications. This section is concluded with a discussion on the choice of existing compression algorithms for our scheme.

2.1 Overview

In our scheme, an image compression algorithm is chosen or modified so that its output can be decomposed into several parts. The number of parts and the information contained in each part depend on the compression algorithm. The encryption process classifies each part into two types—the crucial parts and the remaining parts. The crucial parts are those that by themselves reveal significant information about the original image. On the other hand, the remaining parts yield negligible information if the crucial parts are not known. The crucial parts are encrypted with a secure conventional encryption algorithm such as the International Data Encryption Algorithm (IDEA) proposed by Lai and Massey (1991, 1992). The remaining parts are left unencrypted. The security of our scheme depends heavily on the image compression algorithm used.

2.2 Possible Applications

In addition to the obvious reduction in encryption time, an advantage of our scheme is that it allows us to transmit some parts of the image unencrypted. If only the unimportant parts are transmitted, existing network facilities can be used without the added implementation cost of an encryption layer on top of the existing protocol. Two example applications are presented below.

Distributed image retrieval: The crucial parts may be used to identify an image in a database. If the remaining parts do not have to be encrypted, they can be sent on demand. The collection of the crucial parts of the available images forms a catalogue that is encrypted and stored in each local unit. The catalogue is relatively small, so that it can fit into the limited storage on a portable unit. When a particular image is desired, the user requests an image server to send the remaining parts unencrypted. Without the catalogue, eavesdroppers cannot decode the image. Only subscribed users have the decryption key to access the catalogue, so unauthorized access is effectively denied. The implementation and operating cost of communication encryption can be avoided. Since the remaining parts are not required until a perfect reconstruction of the image is desired, data transmission can be totally eliminated as the user browses through the catalogue. This application is particularly useful in a mobile wireless image retrieval system, in which data transmission is significantly slower than wired networks.

Secure computation: Computing power in a wireless portable unit is often limited, but there may be a powerful server that performs time-consuming computation for the users on the network. Since the server is accessible by any user on the network, the image on which the computation is performed may be obtained by other users. However, if the computation can be carried out using information from the unimportant parts alone, the security of the original image is not compromised.

For example, some image processing computations consider only the individual pixels in the image. These computations include histogram equalization and specification, as well as color and texture statistics computation. Only low-level descriptors of pixels and regions are required in the calculation. The spatial relation of these quantities, stored in the general structure of the image, is not involved. The quadtree method described in Section 4 is one method to decompose an image into its pixel values and its structural component.

2.3 Choice of Compression Algorithms

Any compression algorithm that produces a number of different parts may be used in our scheme, but it is preferred that the crucial parts are relatively small. In this paper, the security of the application of two specific compression algorithms to our scheme is studied.

Many image compression algorithms examine images in the frequency domain. Examples of these algorithms are the JPEG algorithm (Pennebaker and Mitchell, 1993) and those based on discrete wavelet transforms (Mallat, 1989). One way to decompose the output of the JPEG algorithm is to classify the transform coefficients into groups of different frequencies. The low-frequency coefficients give a low-quality approximation of the image, and many high-frequency coefficients are quantized to 0. It is possible that only the low frequency coefficients need to be encrypted. The security of frequency domain decomposition in our scheme is examined in Section 3.

Another class of image compression algorithms compresses images by identifying homogeneous regions in the spatial domain. The output of such an algorithm consists of two parts. One part specifies the location and size of each homogeneous region, and the other part specifies the intensity or color of each region. Typically, the first part reveals significant information about the image, while the second part is useless by itself. It seems plausible to encrypt only the first part. Cohen et al. (1985), Strobach (1991), and Shusterman and Feder (1994) examined an important subclass of spatial domain compression algorithms based on quadtrees. Clarke (1995) showed that quadtree algorithms are computationally simpler than many commonly used compression algorithms including JPEG. The quadtree method offers an efficient way to compress images in a wireless environment. In Section 4, we look at a way to use quadtree compression algorithms in our scheme. The security of our scheme using quadtrees is studied in Section 5.

3 DECOMPOSITION IN THE FREQUENCY DOMAIN

In this section, the security of using image compression algorithms based on frequency domain decomposition is examined. A way to apply such an algorithm to our scheme is described, using JPEG as an example. Finally, the lack of security using frequency domain decomposition is shown.

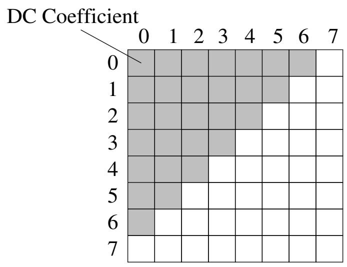


Figure 1 The DCT coefficient matrix and the encrypted coefficients.

3.1 Description of the Method

In a compression algorithm based on frequency domain decomposition, a transformation is applied to an image to obtain its representation in the frequency domain. Some commonly used transformations are the discrete cosine transform (DCT) and discrete wavelet transforms. The coefficients may be divided into groups of different frequencies. As the low-frequency coefficients provide a low-resolution version of the original image, they must be encrypted. However, it may be possible to leave the high-frequency coefficients unencrypted.

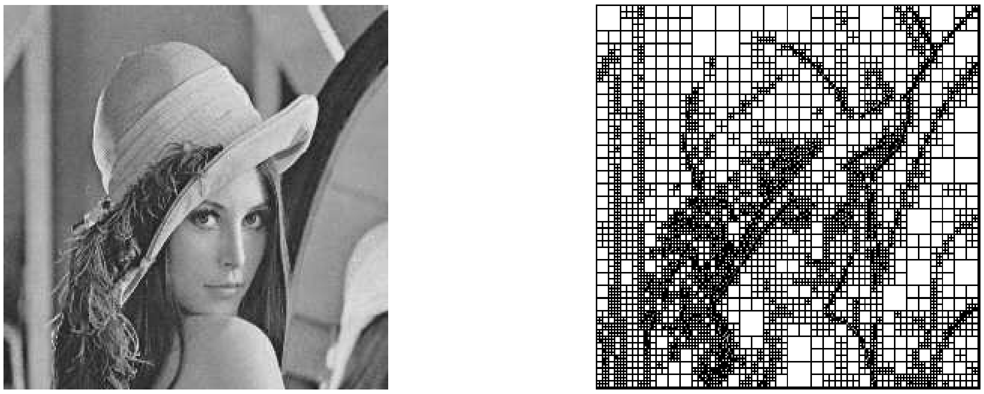
The JPEG algorithm (Pennebaker and Mitchell, 1993) is used as a specific example of this scheme. An image is divided into 8×8 blocks, and DCT is applied to each block. The transform coefficients are then quantized using the normalization matrix suggested by JPEG. The coefficient of the lowest frequency in each block is called the DC coefficient, and all other coefficients are called AC coefficients. Only the JPEG baseline compression algorithm is considered, as it is one of the most commonly used compression algorithms today.

3.2 Security

If only the DC coefficients are encrypted, one can generate a good approximation to the original image by assigning any value to the DC coefficients. This follows from the fact that each DC coefficient only indicates the average intensity of its block. Therefore, some AC coefficients must also be encrypted.

Most of the information about the image is concentrated around the upper-left corner of the coefficient matrix. Let n be the number of rows (or columns) in the matrix, and $C(i, j)$ be the element in the ith row and jth column of the coefficient matrix. Let N be a parameter that controls the number of encrypted coefficients, such that $C(i, j)$ is encrypted if $i + j < N$. If $N = n - 1$, then most of the information about the image block is hidden. This is shown in Figure 1, in which the shaded coefficients are encrypted. This encryption scheme allows us to encrypt slightly fewer than half of the coefficients. The number of encrypted coefficients can be reduced by decreasing the value of N. However, if $N < n + 1$, significant information can be recovered.

Although this approach works well on individual image blocks, it is not secure when applied to every block of the image. Edges in the original image can be seen even if the low-frequency components are unknown. Our experiments show that when only some of the high-frequency coefficients are known, some outlines may appear in the reconstructed image. This is especially apparent for artificially generated images. Essentially, the separation of the coefficients can be viewed as a filter. The unencrypted high-frequency coefficients correspond to the output of a high-pass filter. They give away details about edges where most high-frequency components occur. This also applies to other forms of subband and transform coding schemes, including those based on discrete wavelet transforms. Due to this lack of security, we do not recommend



(a) (b)

Figure 2 The quadtree decomposition of an image. (a) The original image. (b) Its quadtree decomposition.

using our scheme with image compression algorithms based on frequency domain decomposition.

4 DECOMPOSITION IN THE SPATIAL DOMAIN

Another type of decomposition algorithm considers images in the spatial domain. A subclass of these algorithms based on quadtrees was studied by Cohen et al. (1985), Strobach (1991), and Shusterman and Feder (1994). In this section, we examine how the quadtree method is used in our scheme.

Some definitions about quadtrees are first introduced. A **quadtree** is a rooted tree in which every node has 0 or 4 children. A **4-ary tree** is a rooted tree in which every node has 0, 1, 2, 3, or 4 children. A node with no children is called a **leaf**, and all other nodes are called **internal nodes**. The **level** of a node is the number of edges in a path from the root to that node, and the **height** of a tree is the maximum level of all the nodes in the tree.

4.1 Description of the Method

The quadtree method decomposes an image into two parts. The first part is a tree structure that shows the location and size of each homogeneous region. The second part specifies the pixel value at each leaf node of the tree, representing the intensity of the corresponding region. The tree structure alone reveals a significant amount of information about the image. On the other hand, little information can be derived if only the pixel values are known. An example of an image and its quadtree decomposition is shown in Figure 2. The sizes of the tree structures relative to the total compressed data for several test images are shown in Table 1. These images have been compressed using a lossy quadtree algorithm. In our approach, only the tree is encrypted.

In lossless compression, the bit planes of the image are represented by binary images, which are compressed independently using the quadtree method proposed by Cohen et al. (1985). A number of trees and a number of sets of pixel values are obtained. The trees are encrypted together, but the pixel values are unencrypted. Each of the pixel values in this scheme is represented by one bit.

In the case of lossy compression, Strobach (1991) and Shusterman and Feder (1994) proposed

Table 1 The sizes of the output of a lossy quadtree algorithm on several test images.

Image	Total Size	Tree Size	
	Bytes	Bytes	Percentage
Image1	12417	1778	14.3%
Image2	6268	1200	19.1%
Image3	5975	955	16.0%

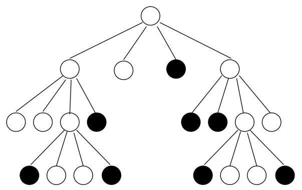


Figure 3 The quadtree of a binary image.

compression algorithms that produce only one tree and one set of pixel values. The number of bits used to represent each of the pixel values depends on the quality desired and the level of the leaf node corresponding to each pixel value.

4.2 Organization of the Pixel Values

The unencrypted pixel values must be organized so that little information can be obtained from them. It is shown in the next section that this organization is crucial to the security of our scheme. The quadtree in Figure 3 is used to illustrate the two formats of the pixel values described below. This quadtree corresponds to a binary image. Black is represented by 1, while white is represented by 0. These formats can easily be generalized to grayscale and color images.

The first organization of pixel values, which we call Format I, is a list of the leaf values from left to right. For example, the pixel values in Figure 3 are encoded as 0010011011110010.

In another organization, Format II, the leaves of the tree are traversed one level at a time, starting from the root. At each level, the pixel values at the leaves are scanned from left to right in the spatial domain one row at a time, starting from the top row. For the tree in Figure 3, if we assume that the four branches represent (from left to right) the NW, SW, SE, and NE subblocks, then the binary image of each level is shown in Figure 4. The output for each level is:

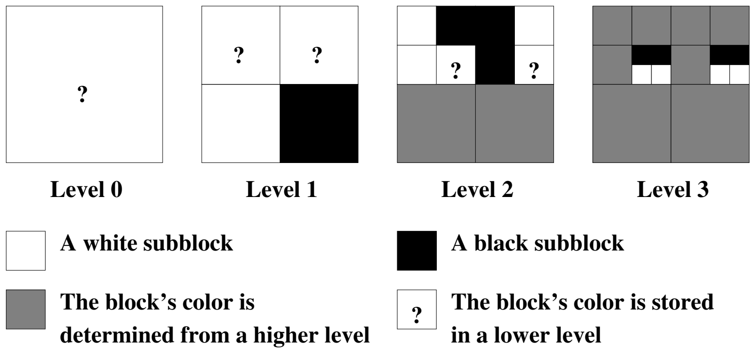


Figure 4 The binary image of each level of the tree in Figure 3.

Level 0: empty
Level 1: 01
Level 2: 011001
Level 3: 11110000

5 SECURITY OF QUADTREE METHODS

Attacks against our scheme generally involve human intervention to determine whether the reconstructed image is correct. Thus, such an attack takes much longer than those against conventional encryption algorithms.

Most encryption schemes are susceptible to brute force attacks. Therefore, it is important that our scheme withstand such an attack. The difficulty of a brute force attack is first discussed. The security in both lossless and lossy compression is then examined. In the following discussion, it is assumed that a secure conventional encryption algorithm has been used to encrypt the tree structure, and hence, the tree structure is effectively hidden. None of the attacks described below rely on breaking the conventional encryption algorithm to recover the tree.

5.1 Brute Force Attack

If the number of bits used to represent each leaf is known and fixed, the number of leaf nodes in the hidden quadtree can be determined from the pixel values. Once the number of leaves is known, the size of the original image can be estimated from typical compression ratios. The quadtree of a $2^n \times 2^n$ image has a maximum height of n. One can generate all possible quadtrees with a specified number of leaves and a specified maximum height, and match each tree with the pixel values. One of the trees must be correct, and the image can be recovered from this tree.

To examine the complexity of this brute force attack, the number of quadtrees given the number of leaf nodes and the maximum height must be computed. Some properties of quadtrees are stated below without proof. The proofs can be generalized from those given by Knuth (1973) for binary trees.

Lemma 1 *A nonempty quadtree has $4k+1$ nodes, where k is a nonnegative integer. In addition, it has k internal nodes, and $3k+1$ leaves.*

Lemma 2 *There is a one-to-one correspondence between quadtrees of height $h+1$ having $4k+1$ nodes and 4-ary trees of height h having k nodes. In other words, the number of quadtrees having $3k+1$ leaves and a maximum height of $h+1$ is the same as the number of 4-ary trees having k nodes and a maximum height of h.*

We are now in position to determine the number of quadtrees given the number of leaves and the maximum height. The following derivation is similar to the one presented by de Bruijn et al. (1972). Let a_{kh} denote the number of 4-ary trees with k nodes and a maximum height of h. By Lemma 2, a_{kh} is also the number of quadtrees having $3k+1$ leaves and a maximum height of $h+1$. Let $g_h(z) = \sum_k a_{kh} z^k$ be the generating function of a_{kh} for a fixed h. Thus, the number of 4-ary trees with k nodes and maximum height h is the coefficient of z^k in $g_h(z)$.

Clearly, $g_0(z) = z$. A 4-ary tree can be formed by adding 0, 1, 2, 3, or 4 subtrees, each with a maximum height $h-1$, to each node, and there are $\binom{4}{i}$ ways to add i subtrees to a node. Thus, the following relation on $g_h(z)$ is obtained:

$$g_h(z) = z \sum_{i=0}^{4} \binom{4}{i} g_{h-1}(z)^i \tag{1}$$

This gives us a way to compute the number of quadtrees:

Theorem 3 *If a_{kh} is calculated according to* (1)*, then the number of quadtrees having $3k+1$ leaves and a maximum height of $h+1$ is a_{kh}.*

The number of trees can be large even with moderate sizes of k and h. For example, there are more than 2^{125} quadtrees when $(k, h) = (60, 4)$. Typical images have $k > 1000$ and $h \geq 8$, and there are many more quadtrees given the two parameters. Exceptions occur when k is small or k is very close to $(4^{h+1} - 1)/3$, indicating either a sparse tree or a tree that is almost full. However, these exceptions occur rarely in typical images. Therefore, the number of possible trees makes this brute force attack infeasible.

We have made the assumption that the number of leaf nodes in the hidden quadtree can be obtained easily. This is only true for lossless compression on binary images. If a grayscale or color image is compressed by a lossless quadtree compression algorithm, several sets of pixel values are obtained. It is difficult to obtain the individual sets when they are concatenated. For lossy compression, the number of bits used to represent each leaf varies. It is very difficult to determine the number of leaf nodes to carry out this attack. Therefore, this brute force attack is not feasible, and our scheme is secure against this attack.

5.2 Security of Lossless Compression

When a lossless quadtree compression algorithm is used in our scheme, only a few significant bit planes are needed to gain important information about the image. Consequently, we can concentrate our efforts in recovering the binary image of only one bit plane.

Format I

When Format I is used to organize the pixel values, one can examine the order in which the pixel values are listed. It is assumed that the four branches of a node represent (from left to right) the NW, SW, SE, and NE subblocks. It is clear that the first pixel value corresponds to the leftmost leaf, and hence the pixel at the NW corner of the image. Similarly, the last pixel value corresponds to the pixel at the NE corner. It follows that two pixels of the image are revealed.

A **run** is a sequence of consecutive identical pixel values. Four or more consecutive pixel values cannot be sibling nodes in the tree; if they were, they would have been merged by the quadtree algorithm. Long runs can be used to obtain information about the image. If a pixel value in the run, p, is a NW leaf in a particular subtree and k is the length of the run starting at p, then a portion of the image can be recovered by the following procedure:

1. Start with p, and write its value (0 or 1) into the NW corner of the block corresponding to the subtree.
2. Set p to the next value in the run. If the run terminates, stop.
3. Write p to the next pixel in the image (using the order NW-SW-SE-NE).
4. If the pixel is written to a NE corner of some quadrant, then the values in the quadrant should have been merged. Determine the number of pixels that are merged by the quadtree compression algorithm. If the number of pixel values merged is 2^{2r}, then move p back by $3r$ values in the run.
5. Go to step 2.

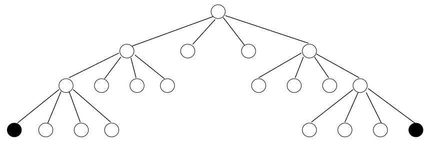


Figure 5 A quadtree corresponding to a run of 14 consecutive pixel values.

The minimum size of the block corresponding to p is $m \times m$, where $m = 2^{\lceil k/3 \rceil - 1}$. Also, the tree structure rooted at the parent of p is highly skewed to the right, indicating a region with uniform pixel values except for a few pixels in its NE corner.

Parallel results can be obtained if one of the pixel values of the run is a NE leaf. Combining with the fact that the first and last pixel values always correspond to the NW and NE corners of the image, we can obtain the minimum sizes of the blocks corresponding to the first and last pixel values.

In general, it is difficult to determine if a pixel value in a long run corresponds to some specific leaf of a subtree. As a result, it is more difficult to exploit the properties of long runs if they do not appear at the beginning or the end of the pixel values. Even if the correspondence is known, neither the exact size nor the spatial location of the corresponding block can be determined.

If a square portion of the image has been obtained, then the quadtree compression algorithm can be applied to this portion. The resulting string of pixel values is a substring of the pixel values for the entire image. Once a match has been found, the pixel values immediately before and after the string must be a NE leaf or a NW leaf, respectively. The known portion of the image can be extended by the above attack.

Some information about the image can be derived from the lengths of long runs without knowing their spatial locations. If the length of a run is k, then the minimum sizes of the blocks represented by the pixel values in the run can be calculated as follows:

1. Set k to the length of run, and i to 0.
2. There are $\min(6, k)$ pixel values corresponding to blocks of minimum size $2^i \times 2^i$.
3. Set $k = k - 6$ and $i = i + 1$.
4. If $k > 0$ then go to step 2. Otherwise, quit.

This procedure gives the minimum sizes of the blocks represented by the pixel values in the run. The only situation in which all the values are the true sizes occurs when a quadtree extends both to the left and the right, but is empty in the middle. This is illustrated in Figure 5, where $k = 14$. Using this method, one can estimate the compression ratio, the size, and the frequency of each intensity level of the original image. If it is known that the minimum size of each block represented by this run is $2^j \times 2^j$, one simply has to set i to j in step 1 of the above procedure.

A drawback of attacks based on long runs is that they can only be used to recover homogeneous regions in images. This method cannot be used to determine nonhomogeneous regions, so the finer details in the image are not revealed. Furthermore, only the minimum sizes of the blocks are known and the tree structures associated with the nonhomogeneous regions cannot be determined. In spite of these drawbacks, one can learn a significant amount of information about the image.

Format II

In order to exploit the properties of long runs in the previous attack, we used the fact that the pixel values of sibling nodes are consecutive in the output of the quadtree compression algorithm. This property does not hold in Format II. Two consecutive pixel values in the output may be very far apart in the tree. The first and last pixel values no longer correspond to particular pixels in the image. In addition, if a portion of the image is known, the output of the compression algorithm on this portion is not a substring of the pixel values for the whole image in general.

The known portion does not help the attack at all. Because of the extra confusion, Format II is much more secure than Format I.

5.3 Security of Lossy Compression

In a lossy quadtree compression algorithm, the pixel values are represented by a variable number of bits. Consequently, it is more difficult to obtain the number of leaf nodes in the hidden quadtree. As we have already seen, Format II is more secure than Format I for lossless compression. It is likely that this is also true for lossy compression, so our discussion concentrates on Format II.

Typically, the number of bits used to encode the pixel values near the root is higher. Shusterman and Feder (1994) used the following bit allocation formula:

$$b_j = \frac{1}{2} \log \left(\frac{{\sigma_j}^2 L_{qt}}{4^j D} \right) \tag{2}$$

where b_j is the number of bits used to store the pixel values at level j of the tree, ${\sigma_j}^2$ is the variance of the leaves at level j, L_{qt} is the total number of leaves in the quadtree, and D is a distortion constant supplied by the user. If there is no leaf on level j of the quadtree, then b_j is set to 0. The distortion constant D is not required for decompression, and it can be discarded after the image is compressed. Each pixel value is rounded and represented by the b_j most significant bits. If the values of b_j are encrypted or hidden, then the individual pixel values cannot be separated and it is difficult to attack this scheme.

An Attack Given the Bit Allocation

Let us assume that the number of bits allocated to the pixel values at each level is known. However, the level at which each leaf is located is still unknown. As a result, the collection of pixel values cannot be separated. Nevertheless, some information about the image can be gained by manipulating Equation 2. The variance at level j can be expressed as

$${\sigma_j}^2 = \frac{4^j D e^{2b_j}}{L_{qt}}.$$

Since L_{qt} and D are constants, the ratios of σ_i to σ_j for $i \neq j$ is

$$\frac{\sigma_i}{\sigma_j} = e^{b_i - b_j} 2^{i-j},$$

provided that $\sigma_j \neq 0$. This set of ratios provides some information about the image, but it does not seem to be significant for recovering the image.

An Attack Given the Individual Pixel Values

If the individual pixel values can be obtained without knowing the bit allocation, they can be grouped by the number of bits allocated to them. The groups are then sorted in descending order. From Equation 2, it is clear that $\{b_j\}$ is most likely a decreasing sequence. Consequently, it is likely that b_j is the length of the pixel values in the jth group in the sorted list. It is now easy to determine the level at which each leaf is located in the tree. The search space in a brute force attack is dramatically reduced. It is possible that $\{b_j\}$ is not a decreasing sequence, but this procedure can be applied to any permutation of the groups. One of the permutations must be correct. There are only $m!$ permutations of the sequence, where m is the number of levels in the tree. For a 1024×1024 image, there are only $11! = 39916800 < 2^{26}$ different permutations of the sequence. In fact, one can assume that $b_0 = 0$ for all nontrivial images. This reduces the number of permutations to $10! = 3628800 < 2^{22}$. It is more efficient to check each possible permutation than to carry out an exhaustive brute force attack.

The above discussion assumes that $\{b_j\}$ is a sequence of distinct nonzero integers. If one of the values in the sequence is zero, then the number of groups obtained is smaller than the number of levels in the tree. If the size of the image is known, an appropriate number of zeroes can be inserted into the sequence and the attack can be applied. However, if there are two integers i and j such that $i \neq j$ and $b_i = b_j$, then we cannot distinguish between the pixel values at levels i and j. The above attack cannot be applied in this case.

To defeat this attack, we must make it difficult to separate the individual pixel values without the corresponding tree. As a result, the pixel values should not be transmitted individually. If it is necessary for an application to separate the pixel values without the tree, then our scheme should not be used.

5.4 Summary

It has been shown that a brute force attack against our scheme is computationally infeasible. In fact, it is more efficient to attack the conventional encryption algorithm used to encrypt the tree structure. The organization of the pixel values must be carefully designed, or too much information is revealed. Consequently, Format II should be used instead of Format I. Finally, attacks on our scheme often depend on the ability to obtain the individual pixel values from the output of the compression algorithm. Thus, our scheme is more secure when lossy compression is performed.

6 CONCLUSION

Our scheme for image compression and encryption has many useful applications in wireless network environments. Provided that the compression algorithm is carefully chosen, our scheme can be as secure as the traditional approach. Frequency domain decomposition is not secure in our scheme. On the other hand, spatial domain decomposition seems promising.

The quadtree compression method was studied as a specific method of spatial domain decomposition. This method works well in both lossless and lossy compression, as long as the pixel values are organized carefully to defeat attacks. If one wishes to use our scheme, compression algorithms based on quadtrees are recommended. Moreover, any quadtree compression algorithm can be used in our scheme, and its analysis remains similar.

Our scheme is a general approach to image compression and encryption, since neither the compression algorithm nor the encryption algorithm has been specified. New compression or encryption algorithms may be used in our scheme if a reasonable level of security is maintained. Our scheme will improve as compression and encryption algorithms improve.

It may also be possible to decompose other types of data, such as video and audio, and classify each part according to the amount of information that can be derived from it. If so, our approach could be applied to these types of data. This may lead to a new way of thinking about data compression and encryption.

REFERENCES

Clarke, R.J. (1995) *Digital Compression of Still Images and Video.*

Cohen, Y., Landy, M. and Pavel, M. (1985) Hierarchical coding of binary images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **7**, 284–298.
de Bruijn, N.G., Knuth, D.E., and Rice, S.O. (1972) The average height of planted plane trees. *Graph Theory and Computing*, 15–22.
Knuth, D.E. (1973) *The Art of Computer Programming: Volume 1, Fundamental Algorithms.* (2nd ed.).
Lai, X. and Massey, J.L. (1991) A proposal for a new block encryption standard. *Advances in Cryptology—EUROCRYPT '90 Proceedings*, 389–404.
Lai, X. (1992) *On the Design and Security of Block Ciphers, volume 1 of ETH Series in Information Processing.*
Mallat, S. (1989) Multifrequency channel decompositions of images and wavelet models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37**, 2091–2110.
Pennebaker, W.B. and Mitchell, J.L. (1993) *JPEG Still Image Data Compression Standard.*
Shusterman, E. and Feder, M. (1994) Image compression via improved quadtree decomposition. *IEEE Transactions on Image Processing*, **3**, 207–215.
Strobach, P. (1991) Quadtree-structured recursive plane decomposition coding of images. *IEEE Transactions on Signal Processing*, **39**, 1380–1397.

7 BIOGRAPHY

About the author—Howard Cheng is currently an undergraduate student in computing science at the University of Alberta, Canada. He plans to continue into graduate studies and his research interests include image processing and algebraic computation.

About the author—Xiaobo Li received the Ph.D. degree in Computer Science from the Michigan State University. His research interests include pattern analysis, computer vision and parallel processing. He is currently an associate professor at the University of Alberta, Edmonton. He served as program co-chair of the Vision Interface '92 conference, and he is an associate editor for *Pattern Recognition*. He is a Senior Member of IEEE.