Journal of Electronic Imaging 7(2), 350–356 (April 1998).

New approach to image encryption

Trees-Juen Chuang
Ja-Chen Lin*
National Chiao Tung University
Department of Computer and Information Science
Hsinchu, Taiwan 30050, Republic of China

Abstract. *A new cryptographic method for encrypting still images is proposed. This method is designed on the basis of the base-switching lossless image compression algorithm. The method therefore simultaneously possesses both image encryption and lossless compression abilities. A given image is first partitioned into nonoverlapping fixed-size subimages, and each subimage will then have its own base value. These subimages are then encoded and encrypted one by one according to the base values. By choosing the function to encrypt the base value, there are $(128!)^t$ [or $(128!)^{3t}$] possible ways to encrypt a gray-scaled (color) image if t layers are used in the encryption system. The theoretical analysis needed to support the proposed encryption method is provided, and the experimental results are also presented.* © *1998 SPIE and IS&T.*
[S1017-9909(98)01202-1]

1 Introduction

Some special images need encryption before transmission or storage. Examples include military and medical images. There are three types of data encryption, namely, block (or transposition),[1] stream (or substitution),[2,3] and product cipher.[1] Various data encryption techniques such as the DES cipher[4,5] (a kind of block cipher) have been applied to image encryption. We review below some recent image encryption techniques. References 6–8 are all based on SCAN language[9,10] and Ref. 11 is a kind of stream cipher based on the quasi-m array,[12] Gold code array,[12] and fast Fourier transform (FFT). In optical systems, the work proposed recently by Refrigior and Javidi[13] is based on random phase encoding. Note that if the random phase mask is not band limited the speckle noise can destroy the optical reconstruction of the encrypted image. The speckle-free kinoform encrypted by smoothed random phase masks is thus required.[14] In TV signal encryption, the scramble method[15] is one of the most commonly used encryption methods, and it encrypts each arbitrary area by means of a kind of block cipher. The tool for image encryption (TIE) system developed by Koch (see Ref. 16 for a short review of the TIE) and the multiresolution image encryption proposed by Macq and Quisquater[17,18] are other new methods in digital TV broadcasting. TIE allows hierarchical access to image data and can create a ''demoversion'' of an image which can be distributed in low resolution. The reconstructed image in high resolution can be delivered and restored by the correct password. The methods proposed by Macq and Quisquater are used in mobile image transmission, and the image is encrypted by permuting details of the multiresolution transform coefficients. Note that such permutation should be executed in order to obtain a ciphered correlated image. Also note that all these kinds of TV signal encryption methods must have at least two properties: transparency and transcodability (Ref. 18 describes these in detail).

For still image encryption, we propose here a new method which can have the function of both lossless image compression and encryption simultaneously. Although Refs. 7 and 11 also use image compression, Ref. 7 is only moderately secure for encrypting quite important images (only $24^{\log n}$ possibilities to encrypt an image of $n\times n$ resolution) and the other is a kind of lossy image compression. Our proposed method provides not only lossless compression but also a high level of security. There are $(128!)^t$ possible ways to encrypt a gray-scaled image, and $(128!)^{3t}$ possible ways to encrypt a color image. Here, the positive integer t represents the number of layers that we used, as will be explained in the last paragraph of Sec. 3.2. In general, $1\leqslant t\leqslant(m\times n/3\times 3)$ if the image is $m\times n$ in size. Note that the number 128! is large as compared with the 10^{16} of DES.[5]

Our method is a kind of stream cipher based on the base-switching (BS) algorithm[19] which is a new lossless image compression algorithm. Because of the compression property, the proposed method can save storage space while increasing security. The remaining part of the paper is organized as follows. In Sec. 2, the overview of the proposed system is presented. In Sec. 3, a review of the BS algorithm is given in Sec. 3.1, followed by the proposed encryption/decryption techniques (Secs. 3.2 and 3.3, respectively). Experimental results, a time complexity analysis, and a comparison with some recent works reported between 1992 and 1995 are illustrated in Sec. 4. The conclusions are stated briefly in Sec. 4.

2 Overview of the Proposed System

As shown in Fig. 1, we first divide the input image (gray-scaled data) into subimages, i.e., blocks, $k\times k$ in size. (Throughout this paper, the subimage size that we use is 3×3 for efficiency of the compression ratio.) The subimages are then processed one by one. In the part of the trans-

*Author to whom all correspondence should be sent.

Paper 96-057 received Oct. 28, 1996; revised manuscript received Sept. 30, 1997; accepted for publication Oct. 7, 1997.
1017-9909/98/$10.00 © 1998 SPIE and IS&T.

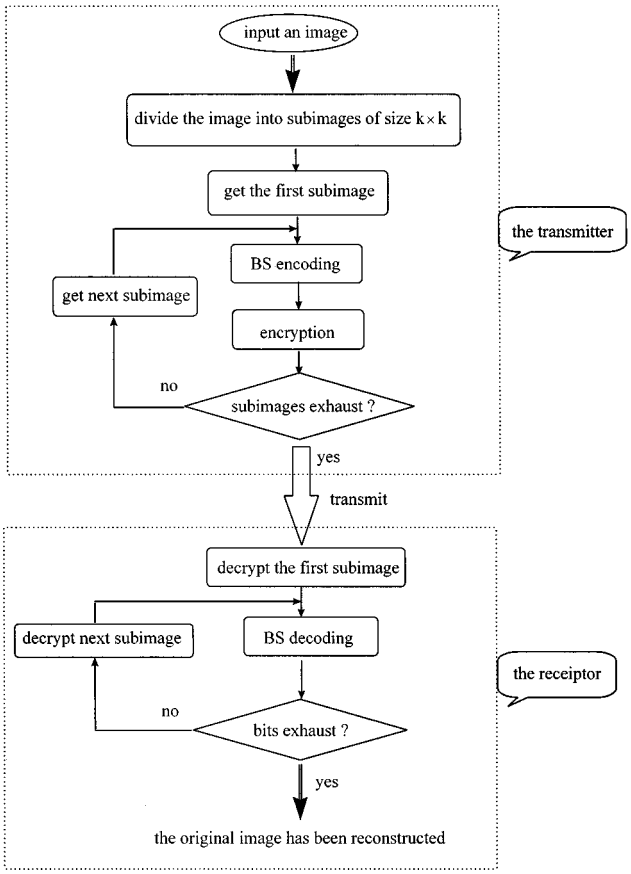


Fig. 1 The system overview.

mitter, for each subimage, we use the BS algorithm to encode the subimage and then encrypt it using the proposed encryption technique. The encoded and encrypted subimage will then be transmitted to the recipient. The subimages are recovered in the recipient end one by one. For each subimage, the recipient first decrypts the subimage and then decodes it using the BS decoding algorithm. Of course, the recipient must know in advance the encrypted keys before decrypting and decoding the subimage received.

In the next section, we sequentially present the three main parts shown in Fig. 1: BS encoding, encryption, and decryption. (An explanation of BS decoding is omitted because it is not essential.) Some definitions and notations are also described and discussed in the next section.

3 The Proposed Method

3.1 *A Review of the BS Transform*

The base-switching transform was originally introduced to compress images.[19] For the reader's benefit, we review BS in this subsection. Let A be a given 3×3 subimage, whose nine gray values are $g_0, g_1, \ldots, g_8$ (see Fig. 2). Define the ''minimum'' m, ''base'' b, and the ''value-reduced subimage'' A' (see Fig. 3) of subimage A by

$$m = \min_{0\leq i\leq 8} g_i, \tag{1}$$

g_0	g_1	g_2
g_3	g_4	g_5
g_6	g_7	g_8

Fig. 2 An arbitrarily given subimage A.

$$b = \max_{0\leq i\leq 8} g_i - \min_{0\leq i\leq 8} g_i + 1, \tag{2}$$

and

$$A'_{3\times3} = A_{3\times3} - m\times I_{3\times3}, \tag{3}$$

respectively. Here, each of the nine elements of $I_{3\times3}$ is 1. Note that (3) means that

$$g'_i = g_i - m \quad \text{for all } i = 0,1,2,\ldots,8. \tag{4}$$

Also note that

$$\min_{0\leq i\leq 8} g'_i = 0$$

and

$$\max_{0\leq i\leq 8} g'_i = b - 1. \tag{5}$$

Therefore, the nine-dimensional vector $A' = (g'_0, g'_1, \ldots, g'_8)$ can be treated as a nine-digit number $(g'_0 g'_1 \cdots g'_8)_b$ in the base-b number system. For convenience, let $V_{3\times3}$ be the collection of all 3×3 subimages A', and B be the base set {1,2,3,... ,256}. Then we define an integer-value function $f: V_{3\times3}\times B \rightarrow$ {non-negative decimal integers} by

g'_0	g'_1	g'_2
g'_3	g'_4	g'_5
g'_6	g'_7	g'_8

Fig. 3 Subimage A'. Each g'_i is $g_i - m$; $i = 0,1,\ldots,8$.

$f(A',b)$

$=$ the decimal integer equivalent to the base-b number $(g'_0 g'_1 \cdots g'_8)_b$

$$=\sum_{i=0}^{8} g'_i \times b^i, \tag{6}$$

$$=\{\cdots[(g'_0 \times b + g'_1)\times b + g'_2]\times b + \cdots\}\times b + g'_8. \tag{7}$$

The following two properties are both proved in Ref. 19.

Property 1. The inequality $f(A',b) < b^9$ always holds.
Property 2. For each base b, and for each given integer λ satisfying $b-1 \leq \lambda \leq \Sigma_{i=0}^{8}(b-1)\times b^i = b^9 - 1$, we can find a unique 3×3 A' such that $f(A',b)=\lambda$.

By Property 1, the number of bits needed to store the decimal integer $f(A',b)$ using a binary number is therefore at most

$$Z_b = \lceil \log_2 b^9 \rceil. \tag{8}$$

When we want to reconstruct $A'=(g'_0, g'_1, \ldots, g'_8)$, all we have to do is to switch that binary (base-2) number to a base-b number $(g'_0 g'_1 \cdots g'_8)_b$. Once A' is known, Eq. (4) implies that A can be obtained quickly by adding m to each pixel value of A'.

3.1.1 *Several possible ways to encode subimage A according to the value of base b*

As stated in (5), for each value-reduced subimage $A'=(g'_0, g'_1, \ldots, g'_8)$, we always have

$$\min\{g'_0, g'_1, \ldots, g'_8\} = 0, \tag{9}$$

and

$$\max\{g'_0, g'_1, \ldots, g'_8\} = b-1. \tag{10}$$

Therefore, at least one of the nine pixels of A' has a gray value of 0, and at least one of the nine pixels of A' has a gray value of $b-1$. As a result, there are at least two ways to encode A. The first way is to store directly

$$\{b;m\} \text{ and a binary equivalent of } (g'_0 g'_1 \cdots g'_8)_b. \tag{11}$$

The second way is to store

$$\{b;m;i_{\min};i_{\max}\}$$

and

$$\{g_i | i \neq i_{\min}, i \neq i_{\max}\}. \tag{12}$$

[Here, $i_{\min} \in \{0,1,\ldots,8\}$ is such that $g'_{i_{\min}}=0$, and $i_{\max} \in \{0,1,\ldots,8\}$ is such that $g'_{i_{\max}}=b-1$. If more than one i in $\{0,1,\ldots,8\}$ have their g'_i value being 0, say, $g'_2=g'_3=g'_5=0$, then use the smallest i as $i_{\min}$ (hence, $i_{\min}=2$ in this case). An analogous statement makes $i_{\max}$ unique.] Because there are $9\times 8=72$ possible combinations of the pair $(i_{\min}, i_{\max})$, we can use $\lceil \log(9\times 8) \rceil = 7$ bits to indicate the positions of the pair $(i_{\min}, i_{\max})$ in the storage system (12). Equation (12) can therefore be simplified to

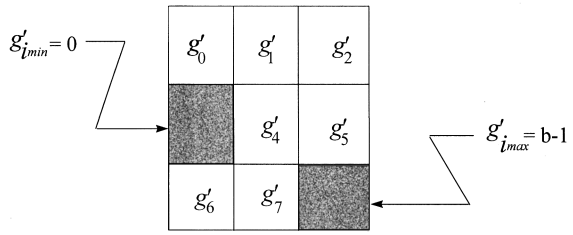


Fig. 4 If the position of $i_{\min}$ and $i_{\max}$ are known, then only seven gray values are needed to be encoded. In this example $i_{\min}=3$ and $i_{\max}=8$ are known.

$$\{b;m;P(i_{\min}, i_{\max})\}, \tag{13}$$

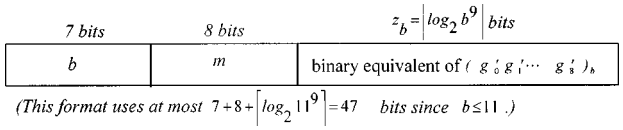
and a binary equivalent of the seven-digit base-b number $(g_i | i \neq i_{\min}, i \neq i_{\max})_b$.

Here, $P(i_{\min}, i_{\max})$ is a 7-bit index used to get the pair $i_{\min}$ and $i_{\max}$. To know when the storage system (13) can save more bits than (11) does, note that first both (13) and (11) need to store b and m; second, (11) needs $\lceil \log_2 b^9 \rceil$ bits to represent a nine-digit number $g'_0 g'_1 \cdots g'_8$ in the base-b number system, whereas (13) needs 7 bits to indicate the location of the $(i_{\min}, i_{\max})$ pair, and another $\lceil \log_2 b^7 \rceil$ bits to encode a seven-digit number $g'_0 g'_1 \cdots g'_8$ (with $g'_{i_{\min}}$ and $g'_{i_{\max}}$ taken away) in the base-b number system. ($g'_{i_{\min}}$ and $g'_{i_{\max}}$ need no storage if we know the position of $i_{\min}$ and $i_{\max}$ (see Fig. 4), because $g'_{i_{\min}}=0$ and $g'_{i_{\max}}=b-1$ always hold according to (9) and (10).) Property 3 is used to compare the storage system (11) and (13). The proof of this property used the fact that $7+\log_2 b^7 > \log_2 b^9$ if and only if $b < 2^{3.5} = 11.314$. The details of this are omitted.

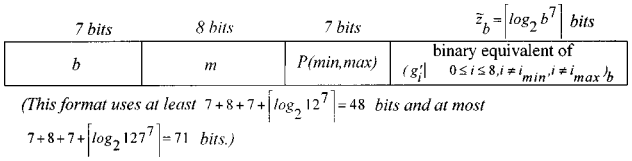
Property 3. Using the storage system (13) is better than using the storage system (11) if and only if $b > 11.314$.

3.1.2 *Formats [to encode subimage $A=(g_0,\ldots,g_8)$]*

Rule 1: If $b \in \{1,2,\ldots,11\}$, then the coding format is

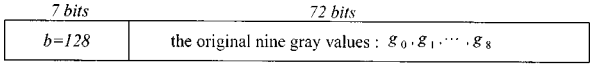


Rule 2: If $b \in \{12,13,\ldots,127\}$, then the coding format is



(This format uses at least $7+8+7+\lceil \log_2 12^7 \rceil = 48$ *bits and at most* $7+8+7+\lceil \log_2 127^7 \rceil = 71$ *bits.)*

Rule 3: If $b \in \{128,129,\ldots,256\}$, then we use a ''dummy'' base $b=128$ and the coding format is



(This format always uses 79 bits.)

In the above, Rule 3 is needed because we cannot use 7 bits to distinguish all 256 possible values of b. Some readers might suggest the use of an 8-bit number to store b so that rule 3 is no longer needed because all b that are greater than 11 can then use rule 2. But we found this would in fact increase the storage space, and the reason is that for most of the images, more than 99% of the subimages have b less than 128 (thus, wasting one more bit in both Rules 1 and 2 is not of merit).

3.2 *Encryption*

After the encoding stage (using Rules 1–3), we could get each encoded subimage and then encrypt it. Our encryption approach is to cipher the base value b for each encoded subimage. Note that unless a cipher ''stealer'' can decipher the base value b, he cannot even know how many of the subsequent bits should be used as the data for the current subimage (see Rules 1–3), not to mention recovery of the subimage. The base value b is in the range $1 \leqslant b \leqslant 128$; we therefore require that the encrypted value $f(b)$ also stays in the range 1–128 for simplicity. Many reported methods could be applied to the encryption of b. The simplest one is the so-called Caesar cipher[20]

$$f(b) \equiv [(b-1+k) \bmod 128] + 1, \tag{14}$$

where k is the key. Beside Eq. (14), there are many other $b \rightarrow f(b)$ mapping functions that can be applied to this encryption system. Note that all these mapping functions are bijective (one to one and onto), and both b and $f(b)$ ($\in Z^+$) are in the range of [1–128]. For example,

$$f(b) \equiv [(k_1(b-1)+k_0) \bmod 128] + 1, \tag{15}$$

where k_0 and k_1 are the keys (affine transformation).[20] Another example is to let

$$f(b) \equiv [(k_0 + k_1(b-1) + k_2(b-1)^2 + \cdots + k_m(b-1)^m) \bmod 128] + 1, \tag{16}$$

where $k_0 - k_m$ are the keys (polynomial transformation[20] of degree m). Some may also use the one-time pad or the Vernam cipher,[21] i.e., let

$$f(b) = [(b-1) \oplus k] + 1, \tag{17}$$

where $k = (k_0 k_1, \cdots k_6)_2$ is a 7-bit key. At any rate, we encrypt an image by mapping the base values b. Because $b \in \{1-128\}$ (we use 7 bits to denote b), and because there are 128! possible functions f that map {1–128} onto {1–128} in a one-to-one manner, our system has 128! possible ways to encrypt a two-dimensional (2D) gray-scaled image. The security level can even be higher using the method stated below. To encrypt the base values $\{b_1, \ldots, b_p, \ldots\}$ of many consecutive subimages by a more powerful function F, we may define

$$\begin{cases} F(b_1) \equiv f_1(b_1) \\ F(b_p) \equiv \{[f_1(b_p) + f_2(b_{p-1})] \bmod 128\} + 1, \end{cases} \tag{18}$$

where f_1 and f_2 are any mapping functions chosen from, say, Eqs. (14)–(17). Note that b_1 is the base value of the first subimage, whereas b_p ($p>1$) is the base value of the pth subimage. The key will be $K = \{k_{f_1}, k_{f_2}\}$ if k_{f_1} and k_{f_2} are the keys of mapping functions f_1 and f_2, respectively. It is easy to see that there are $(128!)^2$ ways to encrypt an image because each f_1 and f_2 has 128! possible choices. [The two-layer scheme (18) can be improved further to a t-layer scheme

$$F(b_p) \equiv \left[\left(\sum_{q=1}^{p} f_q(b_{p-q+1}) \right) \bmod 128 \right] + 1 \quad \text{for } p < t,$$
$$F(b_p) \equiv \left[\left(\sum_{q=1}^{t} f_q(b_{p-q+1}) \right) \bmod 128 \right] + 1 \quad \text{for } p \geqslant t. \tag{19}$$

Of course, t should be smaller than the total number of subimages and the key. $K = \{k_{f_1}, k_{f_2}, \cdots, k_{f_t}\}$ is such that each k_{f_q} is the key of corresponding f_q. Equation (18) could be regarded as a special case of Eq. (19) with $t=2$.] If Eq. (19) is used in our encryption system, then there are $(128!)^t$ possible ways to encrypt a gray-scaled image. The security level is high. Also note that if the image is color our encryption system could be applied three times to the three color components, respectively. We therefore have $(128!)^{3t}$ possible ways to encrypt a color image.

3.3 *Decryption*

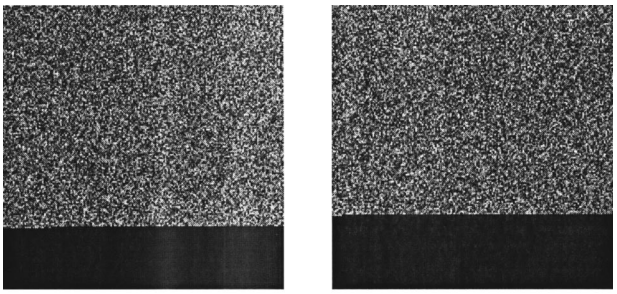
Without the loss of generality, we show below how to decrypt the first subimage and the pth subimage ($p>1$) of an image which has been encoded and encrypted earlier using the techniques introduced in Secs. 3.1 and 3.2. (The remaining subimages can be decrypted similarly.)

We first check the first 7 bits of the received data to get the value of $F(b_1)$ where b_1 is the base value of the first subimage. $b_1 = F^{-1}[F(b_1)]$ can then be known (as explained in the next paragraph), and the BS decoding can then use the base value b_1 to reconstruct (decode) the first subimage.

Because $F(b_1) = [f_1(b_1) \bmod 128] + 1$ according to Eq. (19), b_1 could be solved easily using $b_1 = [f_1^{-1}(F(b_1) - 1)] \bmod 128$ where f_1 is a given mapping function whose key is the known value k_{f_1}. [Some examples of f_1 are Eqs.

(a) (b)

Fig. 5 The original images: (a) Lena and (b) Jet.



(a) (b)

Fig. 7 The incorrectly decrypted images of Figs. 5(a) and 5(b) with $t=2$. [The f_1 and f_2 were used, respectively, in Eqs. (14) and (15).]

(14)–(17).] Note that, because f_1 is bijective, f_1^{-1} always exists. We discuss below how to obtain b_p by Eq. (19) if $p>1$.

Assume that the values of $F(b_p)$ and b_1-b_{p-1} (which have been known by decryption) are all given. We wish to find the value of b_p using the given key $K=\{k_{f_1},k_{f_2},\ldots,k_{f_t}\}$. Note that (19) reads

$$F(b_p)\equiv\left[\left(\sum_{q=1}^{t} f_q(b_{p-q+1})\right)\bmod 128\right]+1$$
$$\equiv\{[f_1(b_p)+f_2(b_{p-1})+f_3(b_{p-2})+\cdots+f_1(b_{p-t+1})]\bmod 128\}+1. \quad (20)$$

Because b_1-b_{p-1} have been decrypted, we can use the key K to compute the value of $C=f_2(b_{p-1})+f_3(b_{p-2})+\cdots+f_t(b_{p-t+1})$. Therefore, Eq. (20) can be rewritten as $F(b_p)\equiv[f_1(b_p)+C]\bmod 128+1$. As a result, $f_1(b_p)\equiv[F(b_p)-1-C]\bmod 128$, i.e.,

$$b_p\equiv f_1^{-1}\{[F(b_p)-1-C]\bmod 128\}. \quad (21)$$

Once the value of b_p is known, the BS decoding system can then be used to reconstruct the pth subimage. Details of the decoding are trivial and, hence, are omitted.

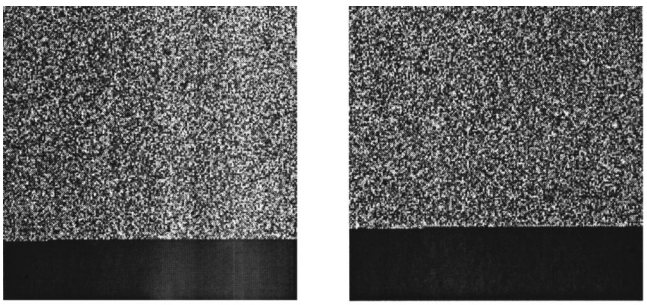
In this section, we present the experimental results obtained by applying the proposed encryption method to two 256-level gray scale images (the ''Lena'' and the ''Jet'' shown in Fig. 5). Figures 6 and 7 show the incorrectly decrypted images of Fig. 5, and Fig. 8 shows the successfully decrypted results. Note that the successfully reconstructed images in Fig. 8 are lossless. Also note that we purposely shifted the decrypted base values b by 1, i.e., $b\rightarrow b+1$, to get Figs. 6(a) and 6(b), and it can be seen that the resulting images are very unrecognizable. We can also see that the image size of the images shown in Figs. 6 and 7 is not even equal to that of the original images shown in Fig. 5 (the last several lines of the incorrectly decrypted images in Figs. 6 and 7 are filled with black). The reason is as explained in the first paragraph of Sec. 3.2.

Table 1 compares the possible decryption ways of some recently published encryption methods and ours. In this paper, the size of each test image is 512×512. Proof that

$$(512\times512)!\ll(128!)^{\lfloor 512/3\rfloor\times\lfloor 512/3\rfloor}$$

is given in the Appendix. Also note that our method has the extra advantage of lossless compression, and the bit rates of the compressed and encrypted images [gray-scaled images (8 bits/pixel) and color images (24 bits/pixel)] are illustrated in Table 2.

We then discuss in this paragraph the time complexity of the proposed method. In compression, 3.44–4.55 clock cycles (the average is 3.94 clock cycles) are required to encode a pixel (a more detailed analysis is presented in Ref. 19). In encryption, if a one-layer (two-layer) encryption scheme is used, three additions/subtractions and one modulus (five additions/subtractions, one multiplication, and one



(a) (b)

Fig. 6 The incorrectly decrypted images of Figs. 5(a) and 5(b) with $t=1$. [The f is that used in Eq. (14).]

(a) (b)

Fig. 8 The successfully decrypted images of Figs. 5(a) Lena and 5(b) Jet.

Table 1 The comparisons of some encryption methods reported recently.

Method	Possible decrypted ways[a]	Compression (lossless/lossy)
(94' ICS)[b]	24^9	Lossless[f]
(93' JEI)[c]	$307008\times(511)\times(511)$	Lossy
(92' PR)[d] and (95' JEI)[e]	$(512\times512)!$	No compression
Our method $\left(t=\left\lfloor\frac{512}{3}\right\rfloor\times\left\lfloor\frac{512}{3}\right\rfloor\right)$	$(128!)^{\lfloor512/3\rfloor\times\lfloor512/3\rfloor}$	Lossless

[a]The test image is gray scaled and 512×512 in size.

[b]Reference 7.

[c]Reference 11.

[d]Reference 6.

[e]Reference 8.

[f]The bit rate of (94'ICS) is about 7.93 bits/pixel for a gray-scaled image.

modulus) are required to encrypt a 3×3-on line subimage, and 7/9 clock cycles (10/9 clock cycles) are therefore needed to encrypt a pixel [the definition of clock cycle under modern very large scale integrated (VLSI) technology can be found in Chap. 2 of Ref. 22]. In other words, the proposed method using a one-layer (two-layer) encryption scheme takes a total of 4.72 (5.05) clock cycles to compress and encrypt a pixel. In general, processing all 512×512 pixels of an image required about 9.29 (9.41) s in a SUN Sparc 10 workstation.

4 Concluding Remarks

A new method for image encryption was proposed in this paper. The method is based on the BS lossless image compression scheme; therefore, both image encryption and lossless compression can be obtained simultaneously. There are $(128!)^t$ [or $(128!)^{3t}$] possible ways to encrypt a gray-scaled (color) image. The fact that decryption without the correct decrypt keys would even fail in knowing the exact image size increases the difficulty for image stealers. Also note that the term $(128!)^t$ [or $(128!)^{3t}$] can also be raised up to be $(256!)^t$ [or $(256!)^{3t}$] if we discard Rule 3 in Sec. 3.1.2 and extend Rule 2 to cover all $b\in\{12,\ldots,256\}$. Of course, this increase in the possible ways to encrypt an image would reduce the image compression ratio somewhat, and the reason for this is given at the end of Sec. 3.1.2.

Note that readers may also combine the proposed method with other image encryption methods based on SCAN language, such as Refs. 6 or 8, to improve security further. Details of this are omitted here.

Table 2 The bit rates (bits/pixel) of the proposed method.

	Scale	
Image	Gray scaled	Color
Lena	5.29	14.72
Jet	5.00	13.11

Appendix

(The proof of $(512\times512)!\ll(128!)^{\lfloor512/3\rfloor\times\lfloor512/3\rfloor}$)

$$
\begin{aligned}
(512\times512)! &= (262144)! \\
&= (10\times26213+14)! \\
&= \left(\prod_{j=1}^{10} j\right) \\
&\quad\times\left(\prod_{j=11}^{20} j\right)\cdots\cdots\left(\prod_{j=262121}^{262130} j\right)\left(\prod_{j=262131}^{262144} j\right) \\
&= \left(\prod_{i=1}^{26213}\ \prod_{j=(i-1)\times10+1}^{i\times10} j\right)\left(\prod_{j=262131}^{262144} j\right)
\end{aligned}
$$

$$
\begin{aligned}
(128!)^{\lfloor512/3\rfloor\times\lfloor512/3\rfloor} &= (128!)^{170\times170} \\
&= (128!)^{28900} \\
&= (128!)^{26213+2687} \\
&= \left(\prod_{i=1}^{26213} 128!\right)(128!)^{2687}
\end{aligned}
$$

since

$$\prod_{j=262131}^{262144} j<10^{76}\ll128!\ll(128!)^{2687}$$

and

$$\prod_{j=(i-1)\times10+1}^{i\times10} j<\prod_{j=262131}^{262144} j\ll(128!)$$

for each $i=1,2,\cdots,26213$, we have

$$(512\times512)!\ll(128!)^{\lfloor512/3\rfloor\times\lfloor512/3\rfloor}.$$

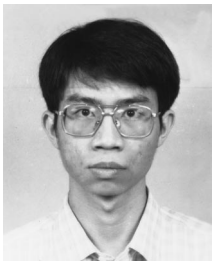
Acknowledgments

This work was supported by the National Science Council, Republic of China, under Contract No. NSC 86-2213-E009-108. The authors thank the referees for the helpful suggestions and clear guidelines for revising the paper.

References

1. D. E. Denning, *Cryptography and Data Security*, Addison–Wesley, Reading, MA (1982).
2. H. Beker and F. Piper, *Cipher System: The Protection of Communications*, Wiley, New York (1982).
3. R. A. Rueppel, *Analysis and Design of Stream Cipher*, Springer-Verlag, Berlin (1986).
4. E. Biham and A. Shamir, ''Differential cryptanalysis of DES-like cryptosystems,'' *Proc. Crypto* 90, Santa Barbara, CA, p. 17 (Aug. 1990).
5. G. B. White, E. A. Fisch, and U. W. Pooch, *Computer System and Network Security*, CRC Press, Boca Raton, FL, Chap. 12 (1996).

6. N. Bourbakis and C. Alexopoulos, ''Picture data encryption using SCAN patterns,'' *Patt. Recog.* **25**(6) (1992).
7. H. K. Chang and J. L. Liu, ''An image encryption scheme based on quadtree compression scheme,'' *Proc. of Int. Computer Symposium (ICS'94), Hsinchu, Taiwan, Republic of China,* pp. 230–237 (Dec. 1994).
8. C. Alexopoulos, N. G. Bourbakis, and N. Ioannou, ''Image encryption method using a class of fractals,'' *J. Electron. Imaging* **4**(3), 251–259 (1995).
9. N. Bourbakis, C. Alexopoulos, and A. Klinger, ''A parallel implementation of the SCAN language,'' *Int. J. Comput. Languages* **14**(4) (1989).
10. M. Blum and S. Micali, ''How to generate cryptographically strong sequences of pseudo-random bits,'' *SIAM* (*Soc. Ind. Appl. Math*) *J. Comput.* **13**(4) (1984).
11. C. J. Kuo, ''Novel image encryption technique and its application in progressive transmission,'' *J. Electron. Imaging* **2**(4), 345–351 (1993).
12. C. J. Kuo and H. B. Rigas, ''2-D quasi *m*-arrays and gold code arrays,'' *IEEE Trans. Inform. Theory* **37**(2) (1991).
13. P. Refregier and B. Javidi, ''Optical image encryption based on input plane and Fourier plane random encoding,'' *Opt. Lett.* **20**(7), 767–769 (1995).
14. L. G. Neto and Y. Sheng, ''Optical implementation of image encryption using random phase encoding,'' *Opt. Eng.* **35**(9), 2459–2463 (1996).
15. J. Slater, ''Scramble: TV signal encryption,'' *Electron. Today* **19**(2), 16–20 (1990).
16. E. Koch, ''TIE (Tool for Image Encryption),'' *http://www.igd.fhg.de/www/hdgdv/sw_catlg/english/tie.html, HdGDV software-TIE, Fraunhofer Institute for Computer Graphics* (1997).
17. B. M. Macq and J.-J. Quisquater, ''Digital image multiresolution encryption,'' *J. Interactive Multimedia Assoc. Intell. Prop. Proj.* **1**(1), 179–186 (1994).
18. B. M. Macq and J.-J. Quisquater, ''Cryptology for digital TV broadcasting,'' *Proc. IEEE* **83**(6), 944–957 (1995).
19. T. J. Chuang and J. C. Lin, ''On the lossless compression of still image,'' *Proc. of Int. Computer Symposium (ICS'96)—On Image Processing and Character Recognition, Kaohsiung, Taiwan, Republic of China,* pp. 121–128 (1996).
20. M. Y. Rhee, *Cryptography and Secure Communications,* McGraw–Hill, Singapore (1994).
21. G. J. Simmons, *Contemporary Cryptology: The Science of Information Integrity,* IEEE Press, New York (1992).
22. J. L. Hennessy and D. A. Patterson, *Computer Architecture—A Quantitative Approach,* Morgan Kaufmann, San Mateo, CA, 3rd printing (1993).



Trees-Juen Chuang received his BS degree from Soochow University, Taiwan, in 1992. Since 1993 he has been studying toward his PhD degree and is currently a PhD candidate in the Computer and Information Science Department of National Chiao Tung University. His recent research interests include pattern recognition, image processing, and image encryption.

Ja-Chen Lin received his BS degree in computer science in 1977 and his MS degree in applied mathematics in 1979, both from National Chiao Tung University, Taiwan. In 1988 he received his PhD degree in mathematics from Purdue University. From 1981 to 1982 he was an instructor at National Chiao Tung University and from 1984 to 1988 he was a graduate instructor at Purdue University. He joined the Department of Computer and Information Science at National Chiao Tung University in August 1988, and is currently a Professor there. His recent research interests include pattern recognition, image processing, and parallel computing. Dr. Lin is a member of the Phi-Tau-Phi Scholastic Honor Society, the Image Processing and Pattern Recognition Society, and the IEEE Computer Society.