Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 8087958, 18 pages
https://doi.org/10.1155/2018/8087958

Research Article

Local Negative Base Transform and Image Scrambling

Gangqiang Xiong,[1] **Shengqian Zheng**,[1] **Jiang Wang**,[1] **Zhanchuan Cai**,[2] **and Dongxu Qi**[3]

[1]*Guangdong Medical University, Dongguan, China*
[2]*Macau University of Science and Technology, Taipa, Macau*
[3]*North China University of Technology, Beijing, China*

Correspondence should be addressed to Gangqiang Xiong; douglasxiong@163.com

Received 5 October 2017; Accepted 29 April 2018; Published 25 June 2018

Academic Editor: Nibaldo Rodríguez

Scrambling transform is an important tool for image encryption and hiding. A new class of scrambling algorithms is obtained by exploiting negative integer as the base of number representation to express the natural numbers. Unlike Arnold transform, the proposed scrambling transform is one-dimensional and nonlinear, and an image can be shuffled by using the proposed transform to rearrange the rows and columns of the image separately or to permute the pixels of the image after scanned into a sequence of pixels; it can be also applied to shuffle certain part region of an image. Firstly, the transformation algorithm for converting nonnegative integers in base B to the corresponding integers in base $-B$ is given in this paper, which is the computational core of scrambling transform and the basis of studying scrambling transform. Then, the three kinds of transforms are introduced, that is, negative base transform (abbreviated as NBT), modular negative base transform (MNBT), and local negative base transform (LNBT) with three parameters, where NBT is an injection and MNBT a surjection and LNBT a bijection. The minimum transform periods of LNBT are calculated for some different values of the three parameters, and the algorithm for calculating the inverse transform of LNBT is given. The image scrambled by LBNT can be recovered by the transform period or the inverse transform. Numerical experiments show that LNBT is an efficient scrambling transform and a strong operation of confusing gray values of pixels in the application of image encryption. Therefore, the proposed transform is a novel tool for information hiding and encryption of two-dimensional image and one-dimensional audio.

1. Introduction

Because of the properties of visibility, intuitive, abundance in information expression and so on, digital images are widely applied in more and more fields such as remote sensing and geographical information, medical image, audiovisual art, radar navigation, copyright, and trademark; therefore, secure transmission and confidential storage of digital images have been gotten more and more attention by government, industry, and academia. Security protection of digital image includes mainly data encryption and information hiding. Image scrambling is a technique of image encryption or auxiliary encryption [1–9], and also an important method of preprocessing and postprocessing in image hiding, sharing, and digital watermarking [7, 10–17]. Particularly in the application of robust digital watermark, scrambling technique is a very effective method against cropping attack by shuffling watermark image so that the pixels are dispersed in the position space of carrier image. The popular image scrambling methods include Arnold transform [1–3, 10–12, 18], Fibonacci transform [7, 10, 13, 14], Peano-Hilbert space filling curve [19], knight's tour [8, 20], Lucas transform [16], magic square transform [9, 17], Gray code [21, 22], cellular automata [15, 23–25], Baker map [4–6, 26], subaffine transform [27], and sampling technology [28].

Most of image scrambling transforms are two-dimensional, such as Arnold transform, baker map, Peano-Hilbert curve, John Conway game, knight's tour, magic square transform, and Fibonacci-Q transform [10]. For two-dimensional scrambling transforms, the popular scrambling transform is Arnold transform, but it requires the sizes of images are the form of $N \times N$ [10]; Fibonacci-Q

transform is the special case of Arnold transforms [10]. Unlike Arnold transform, knight's tour [8] and Peano-Hilbert curve [19], John Conway game [15], and magic square transform [9] are not easy to calculate and are rarely used in the application of image scrambling. In general, if using John Conway game to shuffle images we first need to generate a random matrix of 0 and 1, and thus there are difficulties to restore scrambled images. In fact, Baker map is usually exploited to generate chaotic sequence on the interval $[0, 1) \times [0, 1)$ [4], and if using baker map to shuffle images we must modify the formula of baker map.

One-dimensional scrambling transform can also be used in two-dimensional image encryption and information hiding. Battisti used Fibonacci transform to construct Fibonacci-Haar matrix and then studied digital watermarking and encryption algorithms in Fibonacci-Haar domain [7], while Li and Tsai exploited Fibonacci transform to shuffle watermark information [13, 14]. Gunjal combined Fibonacci transform with Lucas transform to construct Fibonacci-Lucas transform which is applied to shuffle watermark image [16]. Zou investigated a class of generalized Gray code and applied it to scramble color image. Two-dimensional scrambling transform or higher-dimensional scrambling transform [10] generally require that the shape of the region of scrambled image must be rectangular (i.e., the size of $m \times n$) or even square such as Arnold transform. One-dimensional scrambling transform can not only permute a sequence of data, but also permute the pixels in an image region of any shape. In fact, as long as the pixels in the image region are scanned into a sequence in accordance with certain rule, we can exploit one-dimensional scrambling transform to permute these pixels.

In this paper, we use negative number as the base of numeral system to represent the natural numbers and then define negative base transform (abbreviated as NBT) with respect to radix $-B$; construct modular negative base transform (abbreviated as MNBT) with respect to radix $-B$; and modulo B^μ followed by constructing local negative base transform (abbreviated as LNBT) whose radix is $-B$ and modulo B^μ and shift p, where B is an integer greater than 1 and μ, p are natural numbers. In particular, LNBT is a bijective map from the subset $\{p, p+1, \ldots, p+B^\mu-1\}$ of natural numbers to the subset itself. Since LNBT is one-dimensional and nonlinear, we can use it to shuffle one-dimensional audio signal and also to shuffle two-dimensional image. The rest of the paper is organized as follows. Section 2 studies the conversion rules and conversion algorithms from base B integers to the corresponding base $-B$ integers, which is the key of calculating NBT, MNBT, and LNBT. Section 3 gives the definitions of NBT, MNBT, and LNBT and then studies their properties and inverse transforms and last calculates the minimum transformation period of LNBT for some parameter values. Section 4 verifies the scrambling effect of LNBT by simulation experiments, which includes three kinds of scrambling algorithms: scrambling separately along x- and y-direction, scrambling a sequence of pixels after image scanned to a sequence line-by-line or scanned in zigzag order, and then tests the effect against cropping attack in digital watermarking application and the effect of image encryption by shuffling the binary bits of image pixels. Section 5 analyzes the scrambling effect and compares it with other scrambling algorithms; finally, the conclusion is given.

2. Using Negative Base to Express Nonnegative Integers

2.1. Principle of Number Representation with Negative Base. If selecting a negative integer $-B$ ($B > 1$) as the base of number representation, we only need B figures to express all nonnegative integers, that is, we can use the symbols $\{0, 1, 2, \ldots\}$ to express all nonnegative integers [29]. In this paper, for convenience, the notation $(a_n a_{n-1} \cdots a_0)_\beta$ is used to denote an $(n+1)$-digit integer in base β, where β is an integer of absolute value greater than 1, and $a_k = 0, 1, \ldots, |\beta| - 1$, $k = 0, 1, \ldots, n$. For the convenience of description, we also call a_k the kth digit of $(a_n a_{n-1} \cdots a_0)_\beta$. The rest of this paper, we always suppose B is an integer greater than 1. Next, we shall discuss the calculation for conversion from nonnegative integers in base B to integers in base $-B$.

2.2. Conversion Calculation from Base B Integer to Base $-B$ Integer. Assuming that B is an integer greater than 1, a base $-B$ integer can be converted directly to the corresponding decimal integer expressed by the sum of the product which the kth power of $-B$ is multiplied by the kth digit value, as do a base B integer to decimal integer, namely,

$$(a_n a_{n-1} \cdots a_0)_{-B} = a_n \times (-B)^n + a_{n-1} \times (-B)^{n-1} + \cdots a_0 \times (-B)^0. \quad (1)$$

The method of seeking remainder can achieve the conversion from decimal integers to base $-B$ integers [29]. Next, we shall investigate the algorithm for converting base B integers to base $-B$ integers. Although we can apply the method in [29] to convert successfully base B integers to base $-B$ integers after the base B integers being converted to decimal integers, we still need to investigate the method of straightforward conversion of base B integers to base $-B$ integers. There are two main reasons. First, Theorem 7 needs to use the conclusion of Theorem 1, that is, we need to get the relation of the digit capacities between any base B integer and the corresponding base $-B$ integer. Second, sometimes we need to convert base B integers directly to base $-B$ integers. The proposed algorithm can achieve simply the conversion from base B integers to base $-B$ integers; especially the conversion method from binary integers to negabinary integers is simpler. We will consider the issue of the digit capacity of the base $-B$ integer converted from a known base B integer, that is, there is Theorem 1 as follows.

Theorem 1. *Suppose $(a_n a_{n-1} \cdots a_0)_B$ is a positive integer in base B to be converted, where $a_n \neq 0$, while $(x_N x_{N-1} \cdots x_0)_{-B}$ is the converted integer in base $-B$, where $x_N \neq 0$, then $n \leq N \leq n+2$.*

Proof. First, $N \geq n$ holds obviously, followed by N must be even, since $(x_N x_{N-1} \cdots x_0)_{-B} = \sum_{k=0}^{N} x_k (-B)^k < 0$ if N is odd, which contradicts $(a_n a_{n-1} \cdots a_0)_B > 0$.

Next, we prove N is not greater than $n+2$. Denoted by $\{(x_N x_{N-1} \cdots x_0)_{-B}\}_{\min}$ the minimum of $N+1$ digits integers in base $-B$ for all satisfying $x_N \neq 0$. It is obvious that the value of the positive integer as expressed by $(x_N x_{N-1} \cdots x_0)_{-B}$ is smallest only if $x_N = 1$, $x_{N-1} = x_{N-3} = \cdots = x_1 = B-1$, $x_{N-2} = x_{N-4} = \cdots = x_0 = 0$, and there is

$$\begin{aligned} &\{(x_N x_{N-1} \cdots x_0)_{-B}\}_{\min} \\ &\quad = (-B)^N + \sum_{k=0}^{N/2-1} x_{2k+1} (-B)^{2k+1} = \frac{B^N + B}{B+1}. \end{aligned} \tag{2}$$

Assume $N > n+2$, i.e., $N \geq n+3$, since $(a_n a_{n-1} \cdots a_0)_B < B^{n+1}$, we have, from (2)

$$\begin{aligned} &\{(x_N x_{N-1} \cdots x_0)_{-B}\}_{\min} - (a_n a_{n-1} \cdots a_0)_B \\ &> \{(x_N x_{N-1} \cdots x_0)_{-B}\}_{\min} - B^{n+1} = \frac{B^N + B}{B+1} - B^{n+1} \\ &\geq \frac{B^{n+3} - B^{n+2} - B^{n+1} + B}{B+1} \geq \frac{B^{n+2}(B-2) + B}{B+1} > 0. \end{aligned} \tag{3}$$

The above inequality shows that $(a_n a_{n-1} \cdots a_0)_B$ cannot be converted to such a base $-B$ integer of digit capacity greater than $n+2$. □

Theorem 1 shows that $(a_n a_{n-1} \cdots a_0)_B$ can be converted to a base $-B$ integer with $(n+2)$-digit at most, that is, when a_n be converted, there may be carries from the nth digit place to the $(n+1)$th digit place or the $(n+2)$th digit place, but not to the $(n+3)$th digit place. To be more exact, there is the following Corollary 2.

Corollary 2. *Suppose $a_n \neq 0$. If n is even, then $(a_n a_{n-1} \cdots a_0)_B$ can be converted to the forms of $(x_{n+2} x_{n+1} \cdots x_0)_{-B}$ or $(x_n x_{n-1} \cdots x_0)_{-B}$; if n is odd, then $(a_n a_{n-1} \cdots a_0)_B$ can be only converted to the form of $(x_{n+1} x_n \cdots x_0)_{-B}$.*

Proof. It is obvious that $(a_n a_{n-1} \cdots a_0)_B$ cannot be converted to the form of $(x_m x_{m-1} \cdots x_0)_{-B}$ such that $m < n$. Then according to Theorem 1, $(a_n a_{n-1} \cdots a_0)_B$ can be only converted to three forms as follows:

$$(x_{n+2} x_{n+1} \cdots x_0)_{-B}, \tag{4}$$

$$(x_{n+1} x_n \cdots x_0)_{-B}, \tag{5}$$

$$(x_n x_{n-1} \cdots x_0)_{-B}. \tag{6}$$

If n is even, it is clear that the form of (6) is likely to appear, for example, if $a_n \neq 0$, $a_k = 0$ $(k = 0, 1, \ldots, n-1)$, there is $x_n = a_n$, $x_k = 0$ $(k = 0, 1, \ldots, n-1)$ after converted to the corresponding integer in base $-B$. On the other hand, if taking $a_n = B-1$, $a_{n-1} \neq 0$, $a_k = 0$ $(k = 0, 1, \ldots, n-2)$, there is $x_{n+2} = 1$, $x_{n+1} = B-1$, $x_n = 0$, $x_{n-1} = B - a_{n-1}$, $x_k = 0$ $(k = 0, 1, \ldots, n-2)$; this indicates that $(a_n a_{n-1} \cdots a_0)_B$ can be possibly converted to the form of (4). Since it can be deduced from $x_{n+1} \neq 0$ that $(x_{n+1} x_n \cdots x_0)_{-B} < 0$, the form of (5) cannot occur.

If n is odd, it is clear that $(a_n a_{n-1} \cdots a_0)_B$ cannot converted to the forms of (4) and (6); otherwise, there is $(a_n a_{n-1} \cdots a_0)_B < 0$. So $(a_n a_{n-1} \cdots a_0)_B$ is only converted to the form of (5). □

When an integer $(a_n a_{n-1} \cdots a_0)_B$ in base B is converted to the corresponding integer $(x_m x_{m-1} \cdots x_0)_{-B}$ in base $-B$, we suppose the conversion process is from right to left digit-by-digit. Then Corollary 2 shows that, when a_n being converted, there may be two carries need to be carried to the $(n+1)$th digit place and the $(n+2)$th digit place if n is even; otherwise, there is only a carry to the $(n+1)$th digit place. From Corollary 2, we can get the following conversion rules, namely, Corollary 3.

Corollary 3 (conversion rules of base B integer to base $-B$ integer). *Suppose $(a_n a_{n-1} \cdots a_0)_B$ is a base B integer that needs to be converted, where $a_n \neq 0$, and $(x_{n+2} x_{n+1} \cdots x_0)_{-B}$ is the converted base $-B$ integer, where x_{n+2} and x_{n+1} are likely to be zeros. Let $\{b_{n+2}, b_{n+1}, \ldots b_0\}$ be a sequence of carries in the process of conversion, and initial values $b_k = 0$ $(k = 0, 1, \ldots, n+2)$. Then, in accordance with the following rules, base B integer can be correctly converted to the corresponding base $-B$ integer.*

(i) When k is even, if $b_k + a_k \geq B$, then $x_k \leftarrow b_k + a_k - B$, $b_{k+2} \leftarrow 1$, $b_{k+1} \leftarrow B-1$; otherwise, $x_k \leftarrow b_k + a_k$.

(ii) When k is odd, if $b_k - a_k \geq 0$, then $x_k \leftarrow b_k - a_k$; otherwise, $x_k \leftarrow b_k - a_k + B$, $b_{k+1} \leftarrow b_{k+1} + 1$.

The left arrow "$\leftarrow$" in the above denotes that the value of the expression on the right side is assigned to the variable on the left side. In the following, we shall prove that Corollary 3, in fact, only proves that the converted integer in base $-B$ is equal to the original base B integer.

Proof. If k is even and $b_k + a_k \geq B$, there is

$$\begin{aligned} b_k \times (-B)^k + a_k \times B^k = (b_k + a_k - B) \times (-B)^k + (B-1) \\ \times (-B)^{k+1} + 1 \times (-B)^{k+2}. \end{aligned} \tag{7}$$

Corollary 2 indicates $0 \leq b_k \leq B-1$, and from $0 < a_k \leq B-1$, we have $0 \leq b_k + a_k - B < B-1$, i.e., $x_k \leftarrow b_k + a_k - B$. Corollary 2 tells us again that, when a_{k-1} is converted, there is only a carry from the $(k-1)$th digit place to the kth digit place, and when a_{k-2} is converted, there are two carries at most from the $(k-2)$th digit place to the $(k-1)$th digit place and the kth digit place. That is to say, when a_k being converted, the initial values $b_{k+1} = b_{k+2} = 0$. Therefore, it follows from (7) that $b_{k+2} \leftarrow 1$, $b_{k+1} \leftarrow B-1$. If $b_k + a_k < B$, obviously, there is

$$b_k \times (-B)^k + a_k \times B^k = (b_k + a_k) \times (-B)^k, \tag{8}$$

i.e., $x_k \leftarrow b_k + a_k$.

Require: $\{a_n, a_{n-1}, \ldots, a_0\}$ % the sequence of digits of a base B integer
B % the base of the numeral system
Ensure: $\{x_{n+2}, x_{n+1}, \ldots, x_0\}$ % the sequence of digits of the base $-B$ integer
Remark: $\{x_{n+2}, x_{n+1}, \ldots, x_0\}$ % satisfying $(x_{n+2}x_{n+1}\cdots x_0)_{-B} = (a_n a_{n-1}\cdots a_0)_B$
$k = 0$
While $k \le n+2$ **do**
  $x_k = 0$
  $k = k+1$
end while
$k = 0$
while $k \le n$ **do**
  if k is even **then**
    if $x_k + a_k \ge B$ **then**
      $x_k = x_k + a_k - B$
      $x_{k+1} = B - 1$
      $x_{k+2} = 1$
    else $\{x_k + a_k < B\}$
      $x_k = x_k + a_k$
    end if
  else $\{k$ is odd$\}$
    if $x_k - a_k \ge 0$ **then**
      $x_k = x_k - a_k$
    else $\{x_k - a_k < 0\}$
      $x_k = x_k - a_k + B$
      $x_{k+1} = x_{k+1} + 1$
    end if
  end if
  $k = k+1$
end while

Algorithm 1: Converting base B integer to base $-B$ integer $(\{a_n, a_{n-1}, \ldots, a_0\})$.

If k is odd and $b_k - a_k \ge 0$, then $b_k \times (-B)^k + a_k \times B^k = (b_k - a_k) \times (-B)^k$, and the inequality $b_k - a_k < b_k \le B - 1$ indicates that $x_k \leftarrow b_k - a_k$ is true. If $b_k - a_k < 0$, we have

$$b_k \times (-B)^k + a_k \times B^k = (b_k - a_k + B) \times (-B)^k + 1 \times (-B)^{k+1}. \tag{9}$$

The inequality $0 < B - a_k \le b_k - a_k + B \le B - 1$ and (9) imply that $x_k \leftarrow b_k - a_k + B$ and there is a carry 1 from the current digit place to the $(k+1)$th digit place. In addition, it can be known from Corollary 2 that, when a_{k-1} being converted, maybe there exist two carries from the $(k-1)$th digit place to the kth digit place and the $(k+1)$th digit place, namely, b_{k+1} may not be equal to 0 when a_k being converted. Therefore, b_{k+1} need to accumulate the carry 1, i.e., $b_{k+1} \leftarrow b_{k+1} + 1$ is true. □

So we can obtain Algorithm 1 from Corollary 3.

For binary, the conversion process is more concise, and the specific conversion rules are described in the following Corollary 4.

Corollary 4 (conversion rules of binary positive integers to negabinary integers). *Suppose $(a_n a_{n-1}\cdots a_0)_2$ is a binary positive integer to be converted, where $a_n \ne 0$; $(x_{n+2}x_{n+1}\cdots x_0)_{-2}$ is the converted negabinary integer, and the initial values $x_k = 0$ for $k = 0, 1, \ldots, n+2$. Then the following conversion rules hold.*

(i) When k is odd and $a_k = 1$, if $x_k = 0$, then $x_k \leftarrow 1$, $x_{k+1} \leftarrow 1$; otherwise $x_k \leftarrow 0$.

(ii) When k is even and $a_k = 1$, if $x_k = 0$, then $x_k \leftarrow 1$; otherwise $x_k \leftarrow 0$, $x_{k+1} \leftarrow 1$, $x_{k+2} \leftarrow 1$.

(iii) If $a_k = 0$, then the value of x_k keeps changeless.

3. Scrambling Transform Based on Negative Base

Image scrambling, which can change a certain image into another meaningless image, is a special transform from a subset of natural numbers to the subset itself. Rearranging the position of the pixels in the position space of image by permutation on a finite set of the natural numbers can achieve the purpose of scrambling image, such as Arnold transform, baker map, and knight's tour. Next, we define a new class of scrambling transforms based on the principle of number representation in negative bases.

3.1. Negative Base Transform. Unlike the common image scrambling transform, the scrambling transform studied in this paper is one-dimensional, which is such a transform of

Table 1: NBT of the integers between 0 and 15 with parameter −2.

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NBT	0	1	6	7	4	5	26	27	24	25	30	31	28	29	18	19

a certain subset of the natural numbers to the subset itself. First, we define negative base transform in the following.

Definition 5 (negative base transform). For any $x \in \mathbb{N}$, if transform T_{-B} satisfies

$$T_{-B}(x) = \sum_{k=0}^{n} x_k B^k, \tag{10}$$

then $T_{-B} : \mathbb{N} \to \mathbb{N}$ is referred to as negative base transform with respect to base $-B$, abbreviated as NBT, and B is called transformation parameter of NBT, where x_k is the kth digit place of the converted integer $(x_n x_{n-1} \cdots x_0)_{-B}$ in base $-B$ corresponding to x.

From Definition 5, we know that, for computing $T_{-B}(x)$, the natural number x must be converted to the form of $(x_n x_{n-1} \cdots x_0)_{-B}$, and then calculate $T_{-B}(x) = \sum_{k=0}^{n} x_k B^k$. For example, since $7 = (11011)_{-2}$, then $T_{-2}(7) = 2^4 + 2^3 + 2^1 + 2^0 = 27$. Table 1 gives the values of NBT with parameter −2 of the natural numbers from 0 to 15. According to Table 1, we can know that, after the natural numbers between 0 and 15 is transformed by NBT, the order of the transform $T_{-2}(x)$ is already different from the original order of independent variable$x$, and some of $T_{-2}(x)$ are no longer in the set $\{0, 1, 2, \ldots, 15\}$, e.g., $T_{-2}(7) = 27$.

3.2. Modular Negative Base Transform. Assume $\mathbb{N}[\alpha, \beta] = \{x; x \in \mathbb{N} \cap [\alpha, \beta]\}$ denote a subset of the natural numbers from α to β, where $\alpha, \beta \in \mathbb{N}$, $\alpha \le \beta$, and $\mathbb{N}$ denotes the set of natural numbers. In general, Image scrambling transform is a bijective map from a finite set $\mathbb{N}[\alpha_1, \beta_1] \times \mathbb{N}[\alpha_2, \beta_2]$ to $\mathbb{N}[\alpha_1, \beta_1] \times \mathbb{N}[\alpha_2, \beta_2]$. Unfortunately, NBT is an injective map from the set of natural numbers $\mathbb{N}$ to itself, but not surjective, and cannot be used directly in application of image scrambling. Hence we shall make some modifications for NBT so that the modified transform is a bijective map from $\mathbb{N}[\alpha, \beta]$ to $\mathbb{N}[\alpha, \beta]$. According to the transform NBT, we shall introduce another new transform from $\mathbb{N}$ to $\mathbb{N}[0, B^\mu - 1]$ as follows.

Definition 6 (modular negative base transform). For $x \in \mathbb{N}$, if transform $M_{-B,\mu}$ satisfies

$$M_{-B,\mu}(x) = T_{-B}(x) \pmod{B^\mu}, \tag{11}$$

then transform $M_{-B,\mu}$ is called modular negative base transform with respect to base $-B$ and modulo B^μ, abbreviated as MNBT, where μ is an integer greater than 0.

Obviously, MNBT is a surjection from $\mathbb{N}$ to $\mathbb{N}[0, B^\mu - 1]$, but not an injection. In order to make MNBT a bijection, it is necessary to restrict the definition domain of MNBT such that Theorem 7 is true in the following.

Theorem 7. *If the definition domain of MNBT is* $\mathbb{N}[p, p + B^\mu - 1]$, *then MNBT is a bijective map from* $\mathbb{N}[p, p + B^\mu - 1]$ *to* $\mathbb{N}[0, B^\mu - 1]$.

Proof. Suppose $M_{-B,\mu}$ is an MNBT as defined in (11). In fact, we only need to prove that $M_{-B,\mu}$ is injective if $p = 0$, that is, prove that

$$\begin{aligned} &\text{for any} && x_1, x_2 \in \mathbb{N}\left[0, B^\mu - 1\right] \\ &\text{satisfying} && x_1 \ne x_2, \\ &\text{there is} && M_{-B,\mu}(x_1) \ne M_{-B,\mu}(x_2). \end{aligned} \tag{12}$$

Since any integer in $\mathbb{N}[0, B^\mu - 1]$ can only be expressed as a μ-digit integer in base B at most, it follows from Theorem 1 that $x_1$and x_2 only are represented as a $(\mu + 2)$-digit integer in base $-B$ at most. Without loss of generality, we expressed x_1 and x_2 as $(\mu + 2)$-digit integers in base $-B$; if the number of digits is not enough, we fill 0 in front of them, that is,

$$\begin{aligned} x_1 &= \left(x_{1,\mu+1} x_{1,\mu} \cdots x_{1,0}\right)_{-B}, \\ x_2 &= \left(x_{2,\mu+1} x_{2,\mu} \cdots x_{2,0}\right)_{-B}, \end{aligned} \tag{13}$$

where for $j = 1, 2$ and $k = 0, 1, \ldots, \mu + 1$, $x_{j,k}$ is possibly equal to 0. Let

$$\begin{aligned} y_1 &= T_{-B}(x_1) = \sum_{k=0}^{\mu+1} x_{1,k} B^k, \\ y_2 &= T_{-B}(x_2) = \sum_{k=0}^{\mu+1} x_{2,k} B^k. \end{aligned} \tag{14}$$

Assume $M_{-B,\mu}$ is not an injection; there is $x_1 \ne x_2$ such that $M_{-B,\mu}(x_1) = M_{-B,\mu}(x_2)$, namely, $y_1 \equiv y_2 \pmod{B^\mu}$. Therefore, we can infer that

$$x_{1,k} = x_{2,k}, \quad k = 0, 1, \ldots, \mu - 1. \tag{15}$$

Since $x_1 \ne x_2$ implies $y_1 \ne y_2$, then there must be

$$\begin{aligned} &x_{1,\mu} \ne x_{2,\mu} \\ &\text{or } x_{1,\mu+1} \ne x_{2,\mu+1}. \end{aligned} \tag{16}$$

If we can prove that (16) is not true, the fact that $M_{-B,\mu}$ is an injective map is proven, i.e., we complete the proof of the theorem.

Require: x % the nonnegative integer
B, μ, p % the parameters of LNBT
Ensure: $y = f_{-B,\mu,p}(x)$ % the value of LNBT
Remark: y % y satisfying $y = f_{-B,\mu,p}(x)$
$\{x_n x_{n-1} \cdots x_0\}$=**Converting base B integer to base $-B$ integer** $(\{a_n, a_{n-1}, \ldots, a_0\})$
% x satisfying $x = \sum_{k=0}^{n} x_k (-B)^k$
$y = (\sum_{k=0}^{n} x_k B^k \bmod B^\mu) + p$

Algorithm 2: Calculating LNBT (x).

Table 2: The results of LNBT.

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$f_{-2,4,1}(x)$	2	7	8	5	6	11	12	9	10	15	16	13	14	3	4	1

In accordance with (15), there is

$$\begin{aligned} |x_1 - x_2| &= \left| \sum_{k=0}^{\mu+1} x_{1,k} (-B)^k - \sum_{k=0}^{\mu+1} x_{2,k} (-B)^k \right| \\ &= \left| x_{1,\mu+1} (-B)^{\mu+1} + x_{1,\mu} (-B)^\mu - x_{2,\mu+1} (-B)^{\mu+1} \right. \\ &\quad \left. - x_{2,\mu} (-B)^\mu \right| = B^\mu \left| B \left(x_{2,\mu+1} - x_{1,\mu+1} \right) \right. \\ &\quad \left. + \left(x_{1,\mu} - x_{2,\mu} \right) \right|. \end{aligned} \tag{17}$$

If $x_{1,\mu+1} \neq x_{2,\mu+1}$, then, from (17), there is

$$\begin{aligned} |x_1 - x_2| &= B^\mu \left| B \left(x_{2,\mu+1} - x_{1,\mu+1} \right) + \left(x_{1,\mu} - x_{2,\mu} \right) \right| \\ &\geq B^\mu \left(B \left| x_{2,\mu+1} - x_{1,\mu+1} \right| - \left| \left(x_{1,\mu} - x_{2,\mu} \right) \right| \right) \\ &\geq B^\mu \left(B - \left| \left(x_{1,\mu} - x_{2,\mu} \right) \right| \right) \geq B^\mu. \end{aligned} \tag{18}$$

This contradicts obviously $x_1, x_2 \in \mathbb{N}[0, B^\mu - 1]$; thus this shows $x_{1,\mu+1} = x_{2,\mu+1}$.

If $x_{1,\mu} \neq x_{2,\mu}$, then, from (17) and $x_{1,\mu+1} = x_{2,\mu+1}$, there is

$$\begin{aligned} |x_1 - x_2| &= B^\mu \left| B \left(x_{2,\mu+1} - x_{1,\mu+1} \right) + \left(x_{1,\mu} - x_{2,\mu} \right) \right| \\ &= B^\mu \left| x_{1,\mu} - x_{2,\mu} \right| \geq B^\mu. \end{aligned} \tag{19}$$

This also contradicts $x_1, x_2 \in \mathbb{N}[0, B^\mu - 1]$ and shows that $x_{1,\mu} \neq x_{2,\mu}$ is not true. Therefore we have already completed the proof of Theorem 7. □

It is noteworthy that MNBT is a bijective map from $\mathbb{N}[p, p + B^\mu - 1]$ to $\mathbb{N}[0, B^\mu - 1]$, but not from $\mathbb{N}[p, p + B^\mu - 1]$ to the set itself. If taking $p = 0$, then MNBT is a bijective map from $\mathbb{N}[0, B^\mu - 1]$ to $\mathbb{N}[0, B^\mu - 1]$, but in this case, MNBT will map some points to themselves, i.e., the transformation periods of these points are 1, for example, $M_{-B,\mu}(0) = 0$. Thus MNBT cannot be used directly in the application of image scrambling and need to be modified further. In fact, combining MNBT with translational transform with respect to shift p, we can obtain a permutation from $\mathbb{N}[p, p + B^\mu - 1]$ to $\mathbb{N}[p, p + B^\mu - 1]$, which is of course a bijective map.

3.3. Local Negative Base Transform. Local negative base transform can be constructed by combining MNBT with a translational transform with shift p.

Definition 8 (local negative base transform). If the map $f_{-B,\mu,p} : \mathbb{N}[p, p + B^\mu - 1] \to \mathbb{N}[p, p + B^\mu - 1]$ satisfies

$$f_{-B,\mu,p}(x) = M_{-B,\mu}(x) + p, \quad x \in \mathbb{N}\left[p, p + B^\mu - 1\right], \tag{20}$$

then $f_{-B,\mu,p}$ is called local negative base transform with respect to base $-B$, modulo B^μ, and shift p, abbreviated as LNBT.

In Definition 8, $f_{-B,\mu,p}$ is also referred to as LNBT with parameter (B, μ, p). Obviously, if $p = 0$, LNBT is simplified as MNBT. The following corollary is easily established in accordance with Theorem 7.

Corollary 9. *LNBT is a bijective map from $\mathbb{N}[p, p + B^\mu - 1]$ to $\mathbb{N}[p, p + B^\mu - 1]$, i.e., LNBT is a permutation on $\mathbb{N}[p, p + B^\mu - 1]$.*

Corollary 9 shows that LNBT is a scrambling transform from $\mathbb{N}[p, p + B^\mu - 1]$ to itself. Next, we describe the calculation method of LNBT, shown as the Algorithm 2.

For example, if we take $(B, \mu, p) = (2, 4, 1)$, we can calculate easily the values of LNBT according to Algorithm 2, and the calculated results as shown in Table 2, where $f_{-2,4,1}$ in Table 2 is defined as (20).

It is obvious that scrambling transform LNBT can map the subset S of $\mathbb{N}[p, p + B^\mu - 1]$ to $\mathbb{N}[p, p + B^\mu - 1]$; moreover, LNBT can map (scatter) uniformly the entries of the subset S to the set $\mathbb{N}[p, p + B^\mu - 1]$ after many times of LNBT. Figure 1 is the result after 19 times of permutation by LNBT, and it can be seen that the black pixels have been scattered uniformly onto the entire position space of the images; especially, the effect is better in the case of implementing transformation after the image is scanned into one-dimensional sequence (the form of row vector). Generally, as long as B is even and μ is large enough, the scrambling effect is very good. When the pixels are permuted along x-direction, the positional relationship between adjacent pixels at y-direction cannot be changed. Likewise, when permuting the pixels along y-direction, the positional relationship between adjacent pixels at x-direction

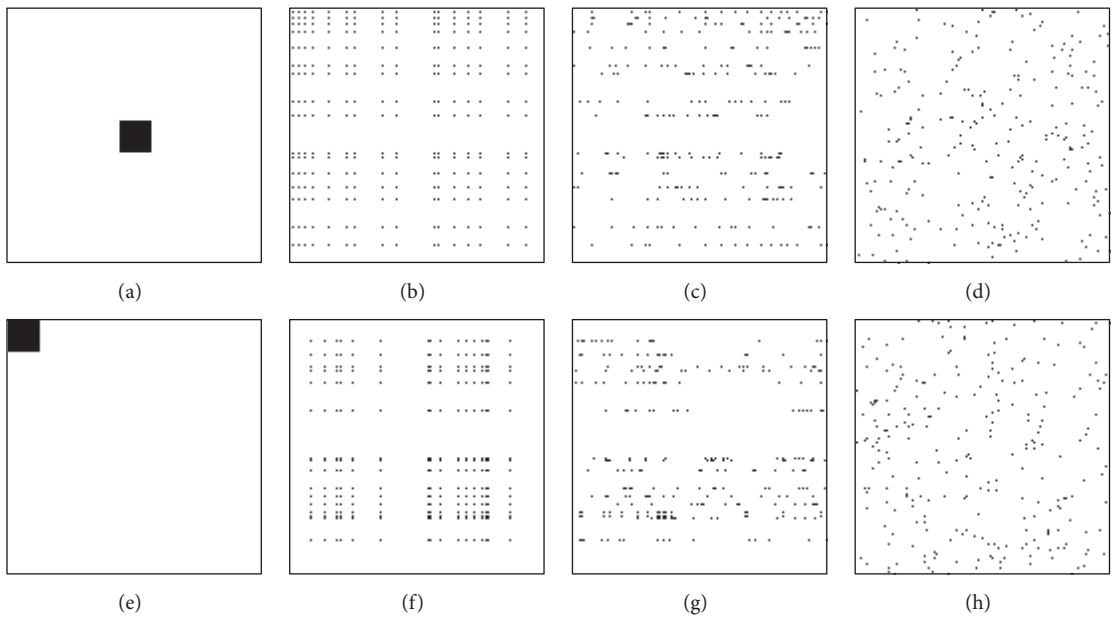


Figure 1: Images of size 128×128 and black area of size 16×16, 19 times of LNBT: (a) black pixels in the center of an image, (e) black pixels in the upper-left of another image; (b) and (f) are the scrambled images of (a) and (e) with parameter $(2, 7, 1)$ along x- and y-direction, respectively, (c) and (g) are the scrambled images of (a) and (e) with parameter $(2, 14, 1)$ after scanned into a sequence line-by-line, and (d) and (h) are the scrambled images of (a) and (e) with parameter $(2, 14, 1)$ after scanned into a sequence in zigzag order.

cannot be changed. Therefore, the pixels shuffled along x- and y-directions separately will result in the black pixels aligned, as shown in Figures 1(b) and 1(f).

3.4. Minimum Transformation Period and Inverse Transformation. In the application of digital watermarking and image encryption, it is often necessary to restore the scrambled image. There are two kinds of methods for restoring scrambled image, that is, by inverse scrambling transform and by minimum transformation period.

3.4.1. Minimum Transformation Period. Like Arnold transform, MNBT and LNBT exist transformation periods, as shown in Theorem 10. In Theorem 10, we use the notations $M^{\lambda_1}_{-B,\mu}(x)$ and $f^{\lambda_2}_{-B,\mu,p}(x)$, which are defined by

$$\begin{aligned} M^{k}_{-B,\mu}(x) &= M_{-B,\mu}\left(M^{k-1}_{-B,\mu}(x)\right), \\ M^{1}_{-B,\mu}(x) &= M_{-B,\mu}(x), \\ f^{k}_{-B,\mu,p}(x) &= f_{-B,\mu,p}\left(f^{k-1}_{-B,\mu,p}(x)\right), \\ f^{1}_{-B,\mu,p}(x) &= f_{-B,\mu,p}(x), \end{aligned} \tag{21}$$

where $M_{-B,\mu}$ and $f_{-B,\mu,p}$ as defined by (11) and (20).

Theorem 10 (period existence theorem for LNBT and MNBT). *There exists a positive integer λ_1 such that $M^{\lambda_1}_{-B,\mu}(x) = x$ for any $x \in \mathbb{N}[0, B^{\mu} - 1]$ and a positive integer λ_2 such that $f^{\lambda_2}_{-B,\mu,p}(x) = x$ for any $x \in \mathbb{N}[p, p + B^{\mu} - 1]$.*

Proof. We only prove that $f_{-B,\mu,p}$ exists the transformation period and can also use the same method to prove $M_{-B,\mu}$. Since the number of entries in the set $\mathbb{N}[p, p+B^{\mu}-1]$ is finite, we only prove that, for any $x \in \mathbb{N}[p, p+B^{\mu}-1]$, there is λ_x such that $f^{\lambda_x}_{-B,\mu,p}(x) = x$; in fact, the transform period of $f_{-B,\mu,p}$ is the common multiple of all λ_x.

Suppose for any positive integer λ there is x_0 such that $f^{\lambda}_{-B,\mu,p}(x_0) \neq x_0$. Let $z_0 = f_{-B,\mu,p}(x_0)$, for any positive integer λ, we have, from the fact that $f_{-B,\mu,p}$ is bijective,

$$\begin{aligned} z_0 = f_{-B,\mu,p}(x_0) &\neq f_{-B,\mu,p}\left(f^{\lambda}_{-B,\mu,p}(x_0)\right) \\ &= f^{\lambda+1}_{-B,\mu,p}(x_0) = f^{\lambda}_{-B,\mu,p}\left(f_{-B,\mu,p}(x_0)\right) \\ &= f^{\lambda}_{-B,\mu,p}(z_0). \end{aligned} \tag{22}$$

On the other hand, let $y_k = f^{k}_{-B,\mu,p}(x_0)$, $k = 1, 2, \ldots, B^{\mu}$, from $x_0, y_k \in \mathbb{N}[p, p + B^{\mu} - 1]$ and pigeonhole principle; we know that there is $i \neq j$ such that $y_i = y_j$. Without loss of generality, assume $j > i$, then from the fact that $f_{-B,\mu,p}$ is a bijective map and $f^{i}_{-B,\mu,p}(x_0) = f^{j}_{-B,\mu,p}(x_0)$, we can conclude $f_{-B,\mu,p}(x_0) = f^{j-i+1}_{-B,\mu,p}(x_0)$, namely, $z_0 = f^{j-i}_{-B,\mu,p}(z_0)$. This contradicts (22). Therefore, there must exist a positive integer λ_x such that $f^{\lambda_x}_{-B,\mu,p}(x) = x$. The proof of this theorem has already completed. □

If using the periodicity of scrambling transform to restore scrambled image, we must find the minimum transformation period of LNBT, but Theorem 10 does not give a calculation

Table 3: Transformation periods on $\mathbb{N}[p, p + B^{\mu} - 1]$.

μ	p	B=2	B=3	B=4	B=5	B=6	B=7	B=8	B=9	B=10
2	1	4	6	16	10	36	14	64	18	100
2	2	2	6	8	10	6	14	32	18	10
2	3	4	2	16	10	12	14	64	6	100
2	4	1	6	2	10	6	14	16	18	10
3	1	8	18	64	50	216	98	512	162	1000
3	2	2	18	32	50	18	98	256	162	50
3	3	8	6	64	50	72	98	512	54	1000
3	4	2	18	4	50	18	98	128	162	50
4	1	16	18	256	50	1296	98	4096	162	10000
4	2	4	18	128	50	108	98	2048	162	500
4	3	16	6	256	50	432	98	4096	54	10000
4	4	4	18	16	50	36	98	1024	162	500
5	1	32	54	1024	250	7776	686	32768	1458	100000
5	2	8	54	512	250	648	686	16384	1458	5000
5	3	32	18	1024	250	2592	686	32768	486	100000
5	4	8	54	64	250	216	686	8192	1458	5000

method of the minimum transform period. In general, it is not easy to determine the minimum transform period of MNBT and LNBT. Table 3 is the minimum transform periods of MNBT and LNBT with

$$(B, \mu, p) \in \{(i, j, k) : i = 2, 3, \ldots, 10;\ \ j = 2, 3, 4, 5;\ \ k = 1, 2, 3, 4\} \tag{23}$$

for $x \in \mathbb{N}[p, p + B^{\mu} - 1]$. For example, for $(B, \mu, p) = (3, 2, 1)$, the minimum transformation periods of all entries on $\mathbb{N}[1, 9]$ are 3 and 6, respectively. Then the minimum transformation period of LNBT with parameter $(B, \mu, p) = (3, 2, 1)$ is 6 as shown Table 3.

Looking at Table 3, we can get the conclusion as shown in Proposition 11.

Proposition 11. *The set $\mathbb{N}[p, p + B^{\mu} - 1]$ is the definition domain of LNBT, where p is nonnegative. For different B and p, there is the following conclusion.*

(i) The minimum transformation periods of LNBT are not greater than B^{μ}. If B=2,4,8 and p is odd, or B is even and $p = 1$, the minimum transformation periods of LNBT is equal to B^{μ}.

(ii) If p is even, the difference between B^{μ} and the minimum transformation period is bigger.

(iii) If $B$is odd, the difference between B^{μ} and the minimum transformation period is also bigger.

Conclusion (i) in Proposition 11 is very important, and in the application of image scrambling, we often take B as an even number and p as an odd number; especially, take $p = 1$ and $B = 2$. In general, the effect is better if the difference between B^{μ} and the minimum transformation period is smaller. In fact, B^{μ} is often a transformation period of LNBT with parameter (B, μ, p), but is not necessarily the minimum transformation period. For example, LNBT with parameter $(-2, 3, 1)$ is a cycle on the set $\mathbb{N}[1, 8]$, namely, $(1\ 2\ 7\ 4\ 5\ 6\ 3\ 8)$, and this shows that the minimum transformation period of LNBT with parameter $(-2, 3, 1)$ is equal to 8. Figure 2 is the experimental results by using minimum transformation period to restore scrambled image, where the parameter values are $(2, 9, 1)$ in (b) and (c).

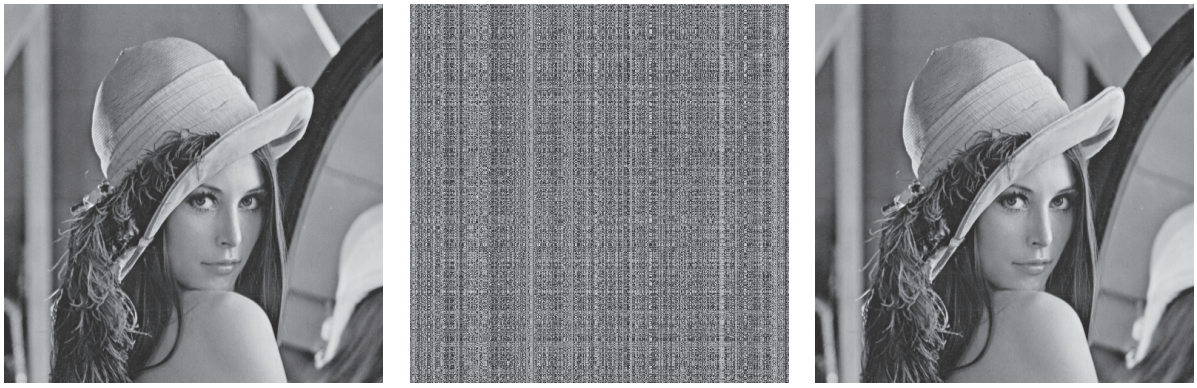
3.4.2. Inverse Transforms of MNBT and LNBT. It is not easy to compute the minimum transformation period of MNBT and LNBT for different parameters (B, μ, p); on the other hand, although the minimum transformation period is sometimes known, but the minimum transformation period is possibly large, so that the computational complexity of restoring scrambled image becomes large. For example, if we use LNBT with parameters $(2, 18, 1)$ to shuffle the pixels of an image of size 512×512 after scanned into a sequence, then the minimum transformation period is 262144. Therefore, the calculation is sometimes difficult by using the periodicity of LNBT to restore scrambled image, while the calculation of restoring scrambled image is more concise and faster by inverse transform of LNBT.

Let $y = T_{-B}(x)$; (10) implies $y = \sum_{k=0}^{n} x_k B^k$, i.e., $y = (x_n x_{n-1} \cdots x_0)_B$ and $x = (x_n x_{n-1} \cdots x_0)_{-B}$. Thus, from (10), the inverse map T_{-B}^{-1} of NBT is obtained directly, namely,

$$x = T_{-B}^{-1}(y) = \sum_{k=0}^{n} x_k (-B)^k, \tag{24}$$

and x_k satisfies the condition $y = (x_n x_{n-1} \cdots x_0)_B$. Equation (24) tells us the process of calculation, which convert y to an integer in base B and then calculate the expression of right side in (24).

Suppose the definition domain of MNBT is $\mathbb{N}[0, B^{\mu} - 1]$. Next, we consider the calculation for the inverse map of $y = M_{-B,\mu}(x)$. According to Theorem 1 and (11), we can conclude that there exists a nonnegative integer k such that $T_{-B}(x) =$



(a) (b) (c)

Figure 2: Restoring scrambled images by the minimum transformation period: (a) is an original images of size 512×512; (b) is the scrambled images of (a) using 22 times of LNBT to shuffle the pixels along x- and y-direction separately; (c) is the restored image using 490 times of LNBT to permute the pixels along x- and y-direction separately from (b).

$kB^{\mu} + y$. From (24), we can get the inverse map $M^{-1}_{-B,\mu}$ of MNBT, namely,

$$x = M^{-1}_{-B,\mu}(y) = \sum_{k=0}^{n} x_k (-B)^k, \tag{25}$$

where x_k satisfies $kB^{\mu} + y = (x_n x_{n-1} \cdots x_0)_B$. In order to compute x in (25), we must first determine the value of k. In fact, the selected k must make the calculated x from (25) satisfy $x \in \mathbb{N}[0, B^{\mu} - 1]$, and from the fact that MNBT is permutation on $\mathbb{N}[0, B^{\mu} - 1]$, we know that k is uniquely determined.

From (25) and (20), we can get the inverse map $f^{-1}_{-B,\mu,p}$ of LNBT, namely,

$$x = f^{-1}_{-B,\mu,p}(y) = \sum_{k=0}^{n} x_k (-B)^k + p, \tag{26}$$

$$kB^{\mu} + y - p = (x_n x_{n-1} \cdots x_0)_B, \tag{27}$$

$$k \text{ satisfying } x \in \mathbb{N}[p, B^{\mu} + p - 1] \text{ in Equation (26)}. \tag{28}$$

Therefore, the inverse transform $x = f^{-1}_{-B,\mu,p}(y)$ can be calculated from (26), (27), and (28), and the specific details are depicted as Algorithm 3.

Figure 3 is the experimental results of image scrambled by LNBT and image restored by inverse LNBT, where the parameter is $(2, 9, 1)$ and the size of image is 512×512. Image (b) is the scrambled image by 19 times of LNBT along x- and y-direction separately and image (c) is the restored image by 19 times of inverse transform of LNBT.

4. Experiments of Image Scrambling

4.1. Scrambling along X- and Y-Direction Separately. Unlike Arnold transform, LNBT is a bijective map from the definition domain $\mathbb{N}[p, p+B^{\mu}-1]$ to definition domain itself, and if using LNBT to shuffle two-dimension image, we can permute the pixels of image along x- and y-direction. Selection of parameter (B, μ, p) is relation to the size of image to be scrambled; suppose the size of image is $M \times N$, we should select $B^{\mu_y}_y = M$ at y-direction and $B^{\mu_x}_x = N$ at x-direction and take p as an odd number. For example, Figure 4(a) is an image of size 216×256; then we can select the parameter $(2, 8, 1)$ at x-direction and $(6, 3, 1)$ at y-direction. Figure 4(e) is another image of size 256×256; then we can select the same parameter $(2, 8, 1)$ both x- and y-direction. From Figure 4, we find that the scrambled pixels are aligned along x- and y-direction, respectively.

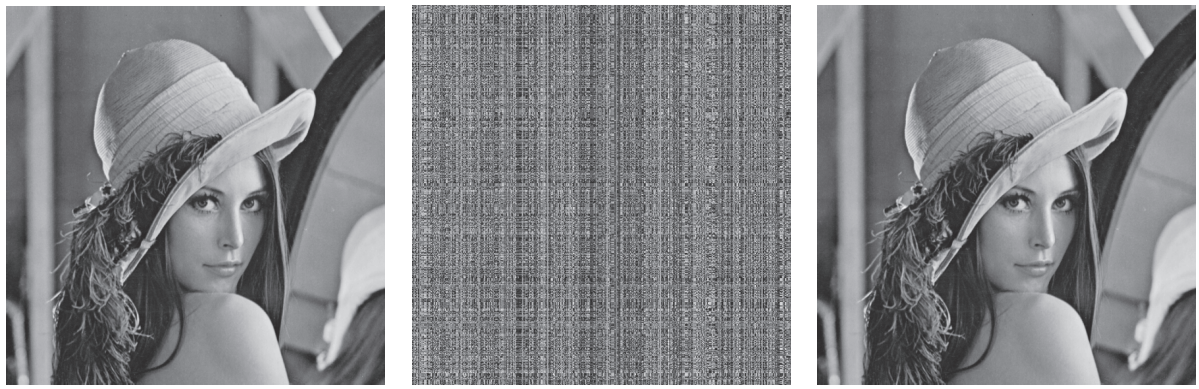
4.2. Scrambling after Scanned into a Sequence

(i) Scan Line-by-Line. In order to eliminate the alignment phenomenon of pixels as described above after image was scrambled along x- and y-direction separately, the easiest method is that we use LNBT to permute the sequence of pixels after two-dimensional image is scanned into a one-dimensional sequence line-by-line. Figure 5(a) is an image of size 216×216, and after being scanned into a one-dimensional vector, the length of the corresponding one-dimension vector is 6^6; thus we select the parameter $(6, 6, 1)$ to shuffle the pixels. Figure 5(e) is another image of size 256×256, and we can select the parameter $(2, 16, 1)$ to shuffle the pixels after being scanned into a one-dimension vector. From Figures 4 and 5, we find that the smaller the parameter B is, the better the scrambling effect is.

(2) Scan in Zigzag Order. After scanning a two-dimension image into a sequence of pixels line-by-line, if taking $B = 2$ we can obtain a good scrambling result by many times of LNBT, as shown in Figure 5(h), but if selecting $B = 6$, the scrambling effect is not very good. In Figure 6, we first use the zigzag scan technique to change a two-dimension image into the corresponding one-dimension vector and then use LNBT to permute this sequence of pixels; last, we use the same zigzag order to arrange the scrambled sequence and

Require: y % non-negative integer
B, μ, p % the parameters of inverse LNBT
Ensure: x % the value of inverse LNBT
Remark: x % x satisfying $x = f^{-1}_{-B,\mu,p}(y)$
$x = -1$
$k = 0$
while $x < p$ or $x > p + B^{\mu} - 1$ **do**
$z = kB^{\mu} + y - p$
$(x_n x_{n-1} \cdots x_0)_B = z$ % converting z into the sequence of base B
$x = \sum_{j=0}^{n} x_j(-B)^j$
$k = k + 1$
7: **end while**

Algorithm 3: Calculating inverse transform of LNBT (y).



(a) (b) (c)

Figure 3: Restoring scrambled image by 19 times of inverse map of LNBT: (a) original image of size 512×512; (b) scrambled image; (c) restored image.

obtain the corresponding scrambled image. Compared with Figure 5, we find the effect of image scrambling becomes better in Figure 6.

4.3. Irregular Region Scrambling of Image. We consider two cases that the size of image cannot express as the form of $B_1^{\mu_1} \times B_2^{\mu_2}$ and the image region to be shuffled is irregular. Since LNBT is a kind of one-dimensional scrambling transform from $\mathbb{N}[p, p + B^{\mu} - 1]$ to $\mathbb{N}[p, p + B^{\mu} - 1]$, if using LNBT to permute two-dimensional image, we must change the image into the corresponding one-dimensional sequence of pixels, and then we permute the sequence of pixels; of course, we can also permute an image along the horizontal and vertical direction separately. Therefore, we can shuffle image region of any shape and images of any size. In the application of part information hiding and encryption of image (e.g., sensitive area scrambling) [30–32], unlike Arnold transform, LNBT can shuffle image region of any shape.

(i) Case 1: Irregular Size. If the size of image cannot be expressed as the form of $B_1^{\mu_1} \times B_2^{\mu_2}$, we cannot use directly LNBT with parameter (B_1, μ_1, p_1) to permute the x-coordinate and with parameter (B_2, μ_2, p_2) to permute the y-coordinate. Therefore, such size of image is called irregular size in this paper. Suppose the size of an image is $M \times N$, and for any positive integers B_1 and μ_1 satisfying $M \neq B_1^{\mu_1}$ or any positive integers B_2 and μ_2 satisfying $N \neq B_2^{\mu_2}$. Without loss of generality, we suppose M cannot be expressed as the form of B^{μ}, then there exist B_x, μ_x, X, and R_x such that $M = X \times B_x^{\mu_x} + R_x$ where $1 \leq R_x < B_x^{\mu_x}$ and thus the set $\mathbb{N}[1, M]$ can be divided into $X + 1$ subsets. For the first X subsets, we select LNBT of parameters (B_x, μ_x, k) to shuffle $\mathbb{N}[\alpha B_x^{\mu_x} + 1, (\alpha + 1)B_x^{\mu_x}]$ where $k = \alpha B_x^{\mu_x} + 1, \alpha = 0, 1, \ldots, X - 1$. For the last subset $\mathbb{N}[XB_x^{\mu_x} + 1, M]$, first, extend the left end of $\mathbb{N}[XB_x^{\mu_x} + 1, M]$ to $\mathbb{N}[M - XB_x^{\mu_x} + 1, M]$; then shuffle the extended subset $\mathbb{N}[M - XB_x^{\mu_x} + 1, M]$ after the first X subsets were shuffled. Figure 7 is the scrambling result of the image of size 720×648. First, permute x-coordinate between 1 and 512 as shown in Figure 7(b), and next, permute x-coordinate between 209 and 720 as shown in Figure 7(c). The results of permuting y-coordinates are shown in Figures 7(d) and 7(e).

(2) Case 2: Irregular Shape. Sometimes we require shuffling a certain region of image [30–32]; in this case, the region to be shuffled may be irregular with respect to size or shape, and therefore it is difficult for two-dimensional scrambling transform to shuffle such irregular region. Unlike Arnold transform, baker map, John Conway game, knight's tour,

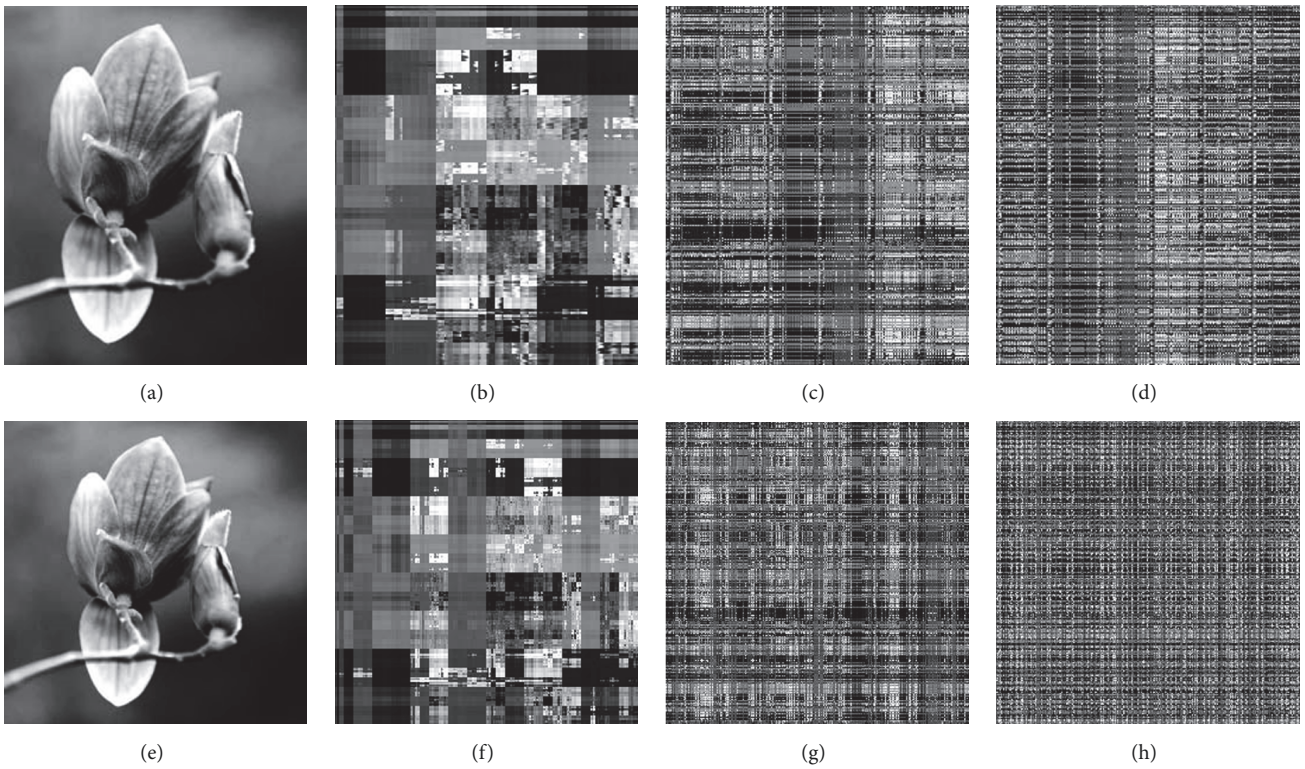


Figure 4: Image scrambling along x- and y-direction separately: (a) original image of size 216×256; (e) original image of size 256×256; (b), (c), and (d) scrambled images from (a) by once, 9 times and 25 times of LNBT, respectively; (f), (g), and (h) from (e) by once, 9 times, and 25 times of LNBT, respectively.

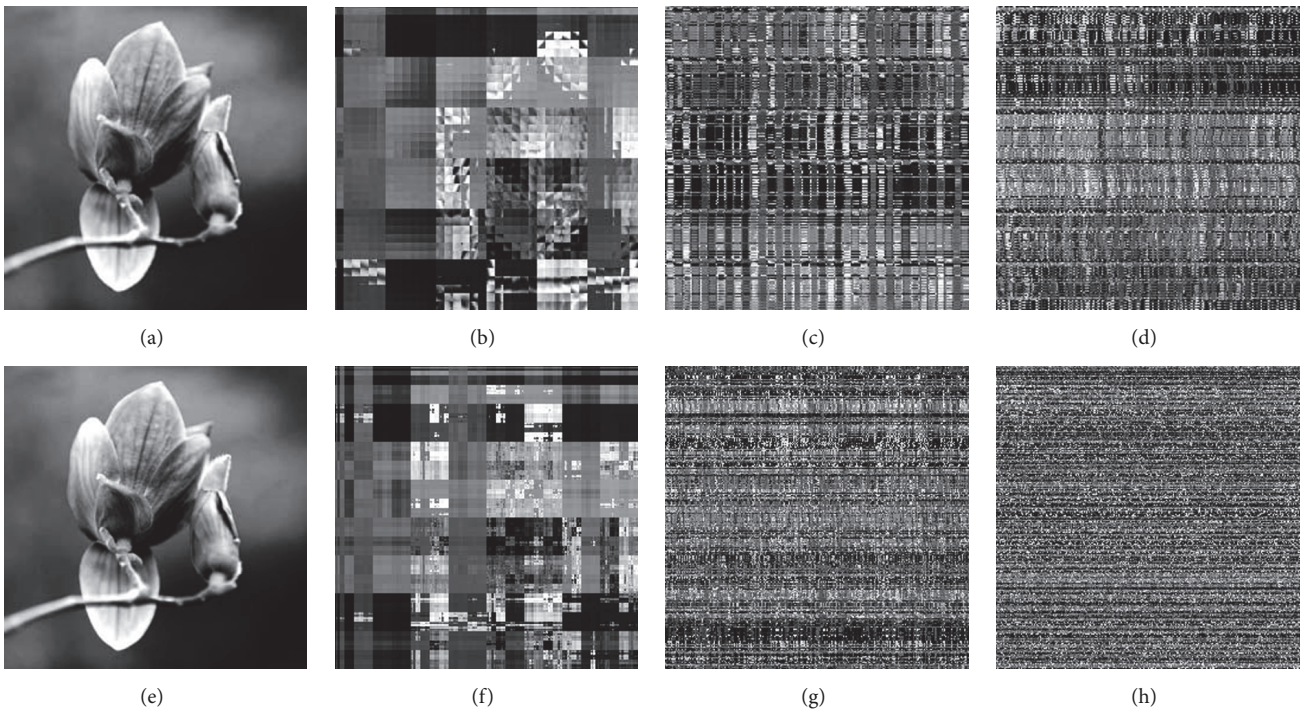


Figure 5: Image scrambling after scanned into a sequence line-by-line: (a) original image of 216×216; (e) original image of 256×256; (b), (c), and (d) scrambled images from (a) by once, 10 times, and 25 times of LNBT with parameter (6,6,1), respectively; (f), (g), and (h) scrambled images of (e) by once, 10 times, and 25 times of LNBT with parameter (2,16,1), respectively.

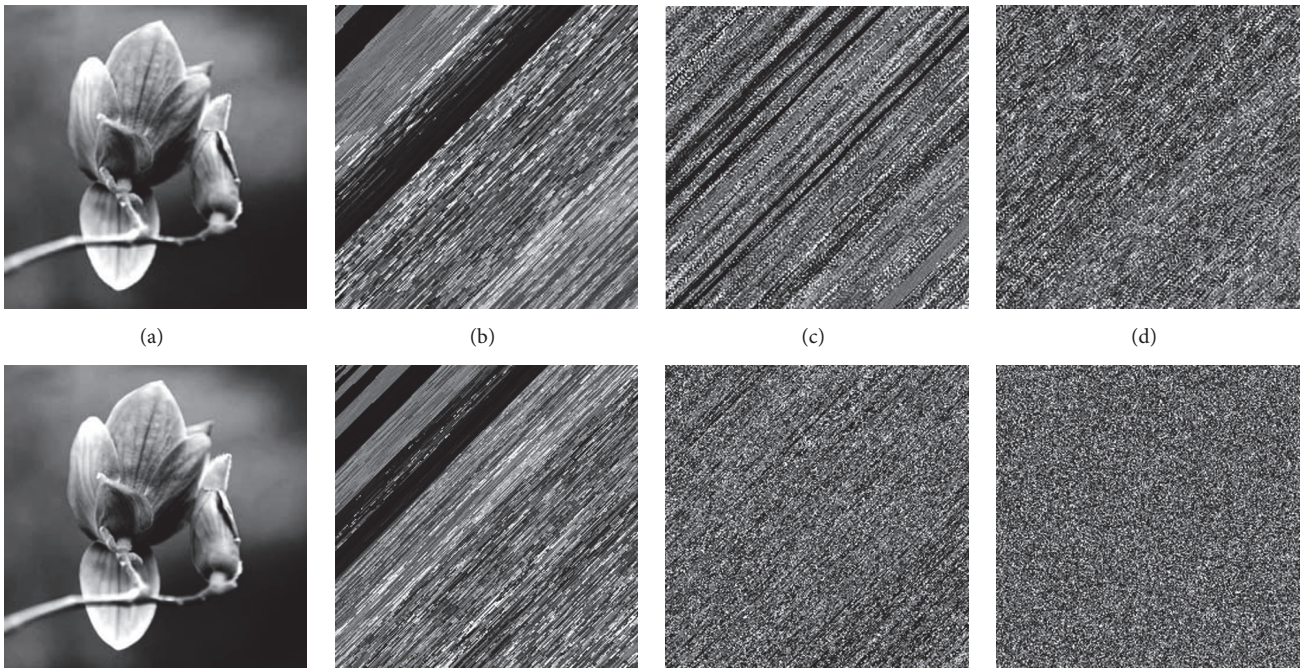


Figure 6: Image scrambling after scanned into a row in zigzag order: (a) original image of size 216×216; (e) original image of size 256×256, (b), (c), and (d) scrambled images of (a) by 1-time, 10-time and 39-time of LNBT with parameter (6,6,1), respectively; (f), (g), and (h) scrambled images of (e) by 1 time, 10 times, and 39 times of LNBT with parameter (2,16,1), respectively.

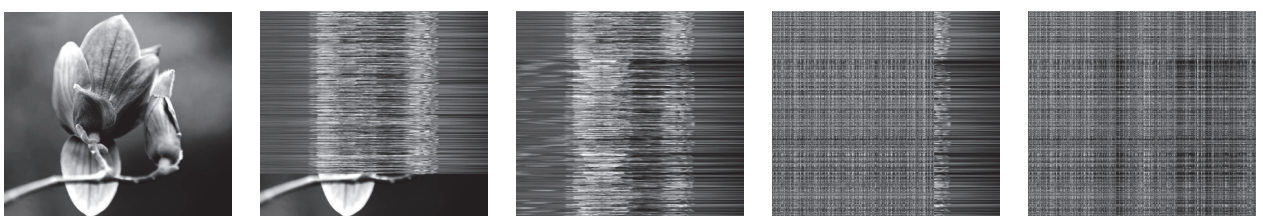


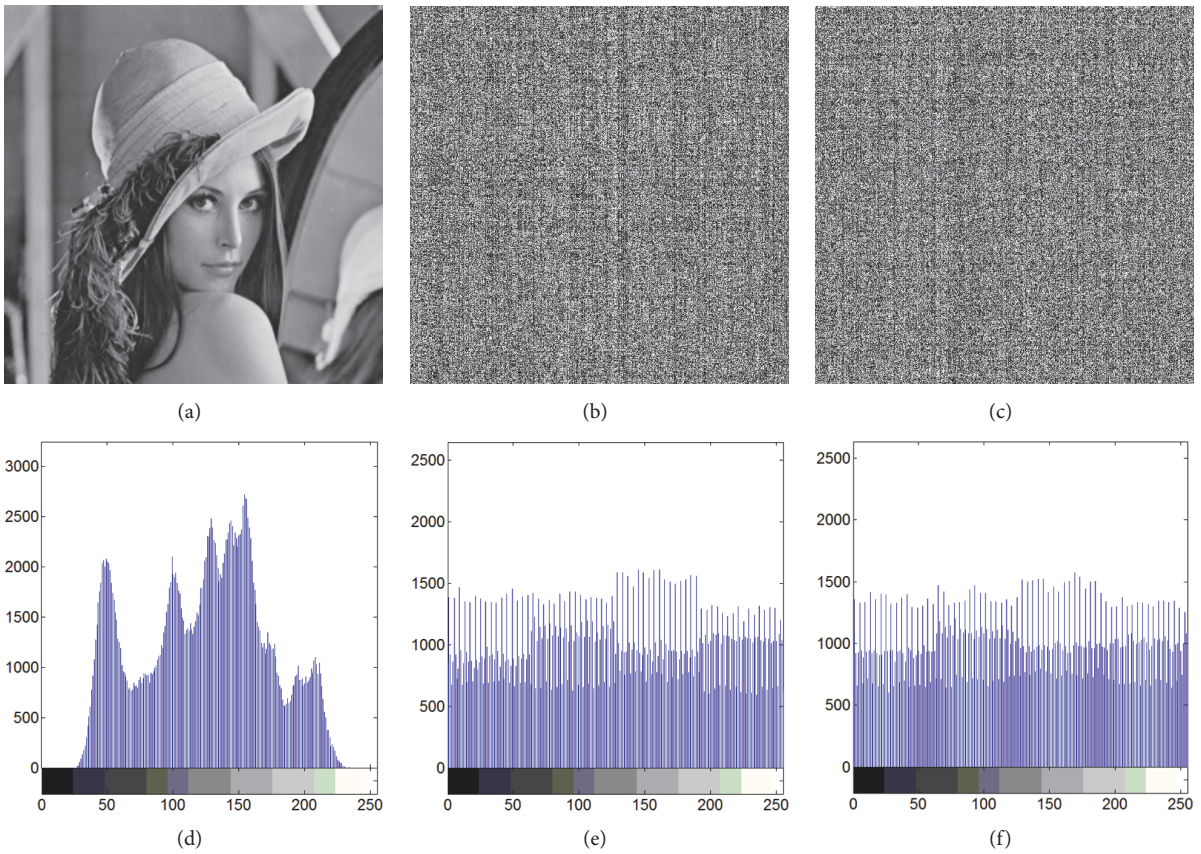
Figure 7: Scrambling image of irregular size by 20 times of LNBT with $(2, 9, p)$: (a) original image of size 720×648; (b) permuting x-coordinate with p=1; (c) permuting x-coordinate with p=209; (d) permuting y-coordinate p=1; (e) permuting y-coordinate with p=137.

magic square transform, and Peano-Hilbert curve, LNBT can be exploited to shuffle region with irregular shape of image, and specific operations are divided into two steps. First, the pixels of the region of image to be scrambled are scanned as one-dimensional sequence; second, use the above scrambling method for irregular size to shuffle the one-dimensional sequence. In Figure 8, we scan the pixels of image region into one-dimensional sequence and then permute these pixels of the sequence. Figure 8 is the results of three shapes by 19 times of LNBT.

4.4. Scrambling on Bit Plane along x- and y-Direction. Permuting the pixel position of an image can only destroy the correlation of adjacent pixels but cannot change the statistical properties of the image, such as histogram and information entropy. This is because scrambling transform based on pixels cannot change the gray value of pixels. If the binary bits of the image pixels are scrambled, it can cause the pixel values to change and thereby can change the statistical properties of the image. If we use binary number of 8 bits to express the value of a pixel, then we can use 0 and 1 to represent a certain image. Therefore the size of a gray image of $M \times N$ is $8 \times M \times N$ and each pixel can be represented by binary block of size 2×4 or 4×2, 1×8, 8×1. For example, if we use a binary block of 2×4 to express a pixel of a gray image of size $M \times N$, we can get a binary matrix of size $(2 \times M) \times (4 \times N)$, and then use LNBT to permute the position of 0 and 1 in the binary matrix corresponding to the gray image. In Figure 9, we first convert original image to a binary matrix where the size of binary block of a pixel is 2×4; next, apply LNBT to shuffle the binary matrix along x- and y-direction separately; finally, we convert the scrambled binary matrix to a scrambled image, i.e., a binary block of size 2×4 is converted to a pixel. From Figure 9, we see that the scrambling transform based on bit

(a) (b) (c)

Figure 8: Scrambling the face region of Lena image, (a) diamond-shaped region, (b) triangular region, and (c) rectangular region.



(a) (b) (c)

(d) (e) (f)

Figure 9: Image scrambling on bit plane by LNBT along x- and y-direction: (a) original image of 512×512l; (b) scrambled image from (a) by 11 times of LNBT at x- and y-direction; (c) scrambled image from (a) by 15 times of LNBT at y-direction and 19 times at x-direction; (d) histogram of (a); (e) histogram of (b); (f) histogram of (c).

plane not only makes the scrambled image meaningless but also changes the statistical properties of the original image. Thus this shows that LNBT is also a tool of image encryption.

4.5. Digital Watermarking against Cropping Attack. In the robust watermarking of digital images, image scrambling is a powerful method for resisting cropping attack. LSB (least significant bit) algorithm is a simple digital watermarking technology, but as a large amount of data hiding method, LSB has a significant position. Figure 10 is the experimental result of combining LNBT with LSB algorithm. The implementation process is as follows. First, use 19 times of LNBT to shuffle the carrier image, then embed the bits of the watermark image into the least significant bits of the scrambled carrier image, and finally, use the inverse transform of LNBT to obtain the

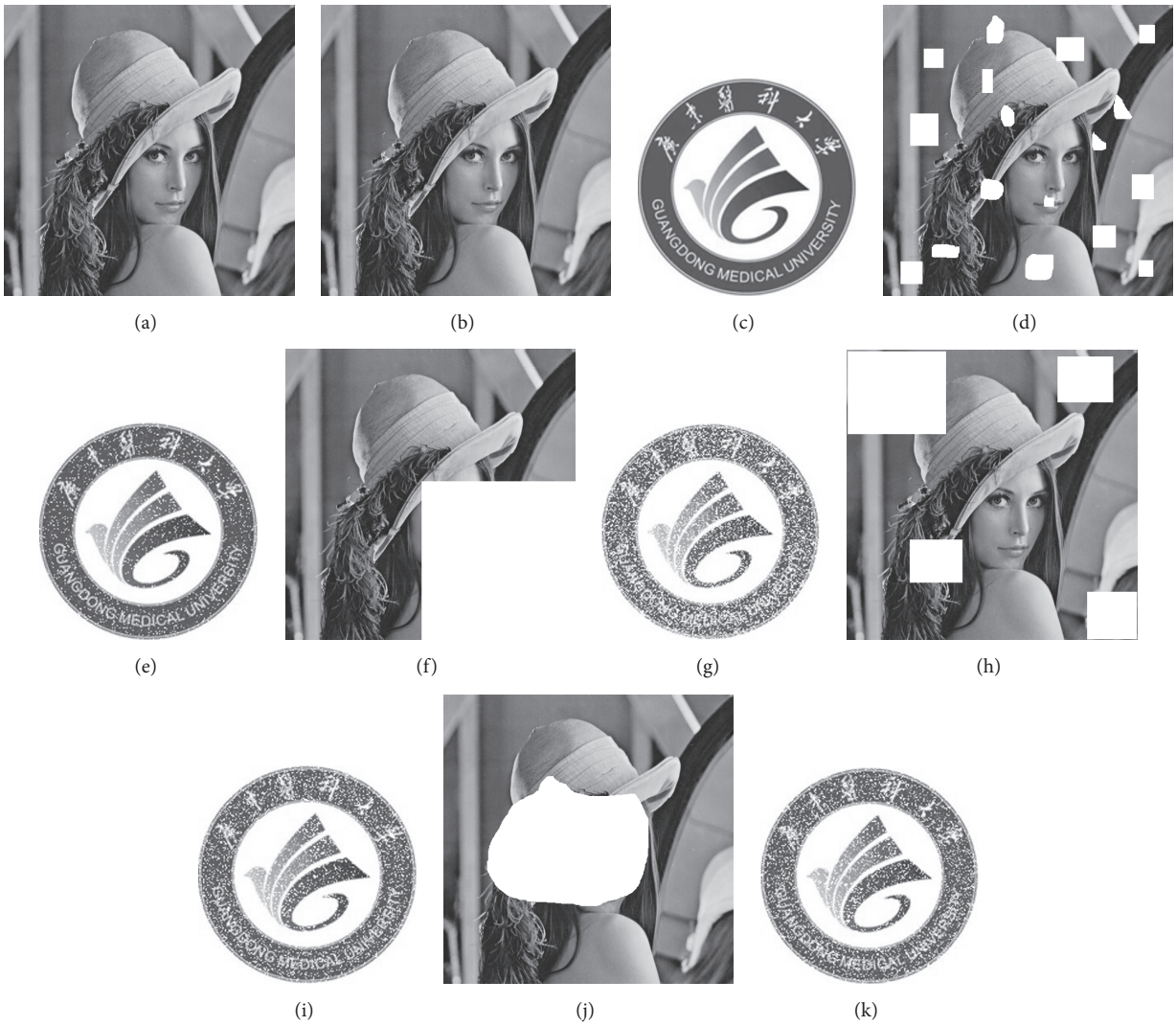


Figure 10: Watermarking experiment based on LNBT and LSB: (a) original image; (b) watermarked image; (c) watermark image, (d), (f), (h), and (j) attacked images by cropping; (e), (g), (j), and (k) watermarks extracted from the attacked watermarked images.

watermarked image (b). Images (e), (g), (i), and (k) are the extracted watermarks from the watermarked imaged after attacked by cropping, and Figure 10 shows that LNBT is effective against cropping attack.

5. Analysis and Conclusions

5.1. Correlations between Adjacent Pixels. The goal of image scrambling is to remove the correlation between the pixels of a plaintext image, and therefore the correlation value between adjacent pixels of image is an important index for evaluating the performance of the image scrambling algorithm. Assuming that x_n and y_n are two sequences consisted of horizontal or vertical or diagonally adjacent pixels, their correlation value can be calculated by [6]

$$Corr = \frac{\left|N\sum_{i=1}^{N}(x_i \times y_i) - \sum_{i=1}^{N} x_i \times \sum_{i=1}^{N} y_i\right|}{\sqrt{\left(N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2\right) \times \left(N\sum_{i=1}^{N} y_i^2 - \left(\sum_{i=1}^{N} y_i\right)^2\right)}} \quad (29)$$

where N is the length of sequences. Obviously, if the correlation value is close to 1, it indicates that there is a higher correlation between adjacent pixels. If the correlation value is close to 0, then the two adjacent pixel sequences have a lower correlation. Therefore, the smaller the correlation value is, the better the performance of the scrambling algorithm is.

To measure the performance of LNBT scrambling algorithm by using the correlation values of neighboring pixel sequences. First, we use other common scrambling algorithms and the proposed algorithm in this paper to perform a 32-round scrambling transformation on the Lena image.

Table 4: GDD and correlation values.

Method name	rounds	GDD	Correlation values		
			Horizontal	Vertical	Diagonal
Original image	\	\	0.9273	0.9590	0.9062
2D Arnold [10]	6	0.8995	0.1239	0.06369	0.0003
Discrete Baker map [26]	11	0.8972	0.0696	0.0384	0.0483
Sub-affine [27]	13	0.8985	0.0807	0.0921	0.0071
CA [15]	13	0.8908	0.0114	0.0270	0.0080
Fibonacci-Q [10]	32	0.7280	0.2968	0.9570	0.3012
Hilbert curve [19]	19	0.8910	0.0083	0.0013	0.0146
Magic cube [9]	27	0.8959	0.0670	0.0514	0.1107
2×2 sampling [28]	1*	0.8957	0.1225	0.0352	0.0770
LNBT-1	32*	0.8864	0.0749	0.1456	0.0431
LNBT-2	32	0.8884	0.0323	0.0170	0.0016
LNBT-3	24	0.8909	0.00005	0.0227	0.0057

∗Remark: visually, the scrambling performance of the third round of 2x2 sampling transform is best, and the 29th round of LNBT-1 has the best scrambling effect. Figures 12(i) and 12(j) are the scrambled images of the third round of 2x2 sampling transform and of the 29th round of LNBT-1, respectively.

Then pick out the largest value of GDD (see Section 5.2) in the 32-round transforms as the last results of each scrambling algorithm for comparison. Finally, we randomly select 5000 pairs of horizontally adjacent pixels, vertically adjacent pixels, and diagonally adjacent pixels and then calculate the correlation values of the neighboring pixel sequences of the three directions in the original image and the correlation values of the three directions in the scrambled images, respectively. Columns 4, 5, and 6 of Table 4 are correlation values of adjacent pixel sequences of the three directions in the original image and in the scrambled images, where LNBT-1 denotes the LNBT scrambling algorithm of permuting separately along x- and y-direction; LNBT-2 denotes the LNBT scrambling algorithm after scanned line-by-line; LNBT-3 denotes the LNBT scrambling algorithm after scanned in zigzag order. We can know from Table 4 that the LNBT scrambling transform can successfully remove the correlation between adjacent pixels. Figures 11(a), 11(b), and 11(c) display the gray-scale distributions of 5000 pairs of horizontally adjacent pixels, of vertically adjacent pixels, and of diagonally adjacent pixels, respectively, which are selected randomly from original image. (d), (e), and (f) are the gray-scale distributions of an adjacent pixel sequence pairs for horizontal, vertical and diagonal directions in the scrambled image using LNBT-1; (g), (h), and (i) show the gray distributions of an adjacent pixel sequence pairs for the three directions in the scrambled image using LNBT-2; (j), (k), and (l) are the gray-scale distributions using LNBT-3. The results of Table 4 and Figure 11 show that LNBT can successfully remove the correlation between adjacent pixels of plaintext image

5.2. Scrambling Degree. Unlike correlation values, scrambling degree can be used to evaluate the correlation of adjacent pixels of a whole scrambled image. If a scrambling algorithm can obtain a large scrambling degree, the algorithm can achieve high confusion and diffusion properties. Image scrambling degree is determined by the gray difference between adjacent pixels [25]. Assuming that $p(i, j)$ is the gray value of the pixel at position (i, j), the gray difference of pixel $p(i, j)$ can be calculated using [23–25]

$$GD(i,j) = \frac{1}{4} \sum_{s,t\in\{(i-1,j),(i+1,j),(i,j-1),(i,j+1)\}} \left(p(i,j) - p(s,t)\right)^2, \quad (30)$$

where $i = 2, 3, \ldots, M-1$, $j = 2, 3, \ldots, N-1$, and the image size is $M \times N$. We use $E(GD)$ to represent the average gray-scale difference of whole image; obviously, $E(GD)$ can be calculated using

$$E(GD) = \frac{1}{(M-2)(N-2)} \sum_{i=2}^{M-1} \sum_{j=2}^{N-1} GD(i,j). \quad (31)$$

Assume that $E(GD)$ represents the average gray-scale difference of plaintext image and $E'(GD)$ represents the average gray-scale of the scrambled image. Then, according to paper [25], image scrambling degree can be calculated as follows:

$$GDD = \frac{E'(GD) - E(GD)}{E'(GD) + E(GD)}. \quad (32)$$

From (32), it can be seen that the value of GDD is between -1 and 1, and the larger the value of GDD is, the better the performance of the scrambling algorithm is. The data in the third column of Table 4 is the maximum value in 32-round transform for each scrambling algorithm. The remark below Table 4 shows that GDD does not accurately evaluate the performance of a scrambling algorithm, and sometimes there is a large visual difference. Figure 12 is the scrambled images using scrambling algorithms to shuffle Lena image. The scrambling effect of Figures 12(e), 12(g), 12(k), and

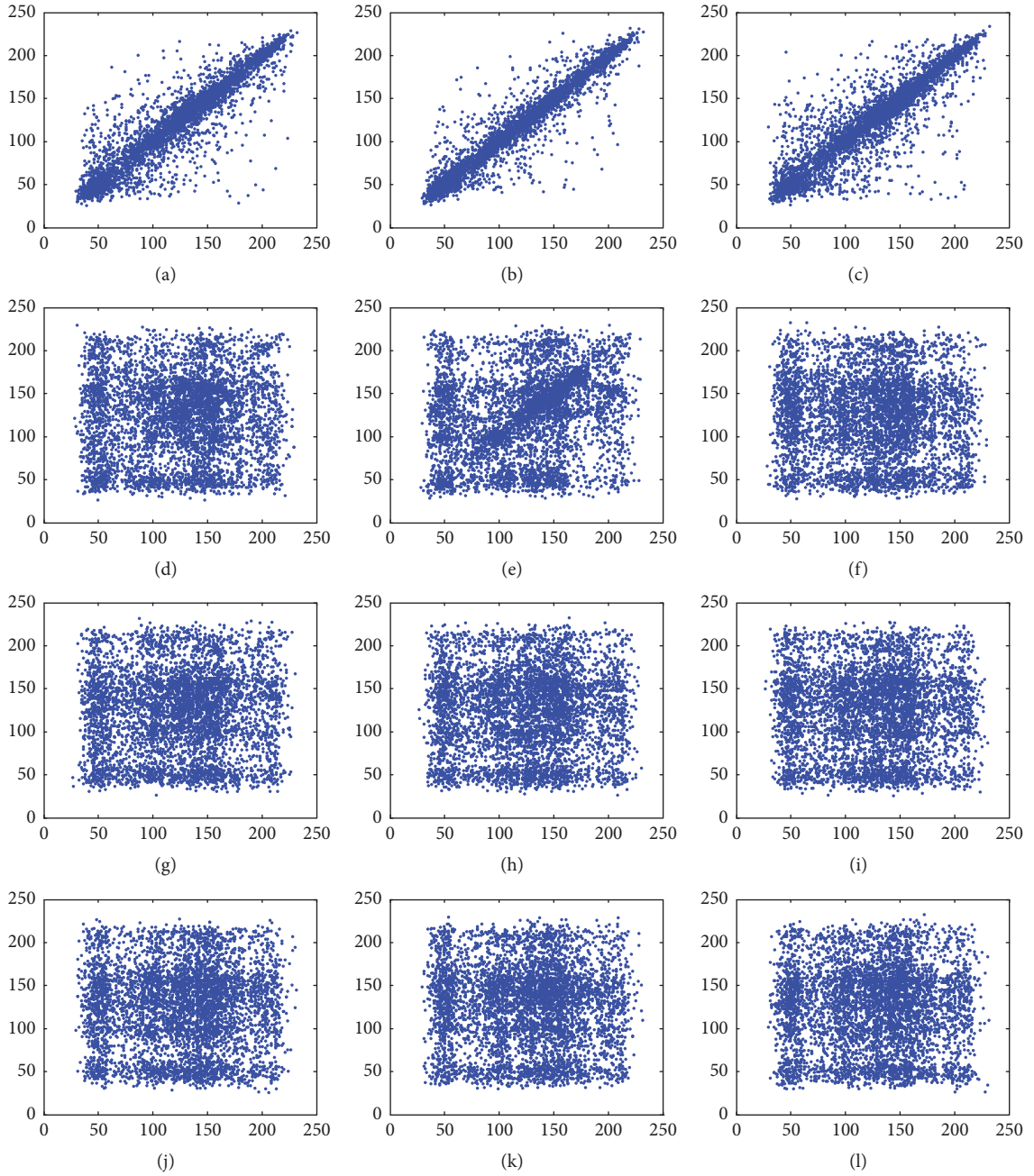


Figure 11: Gray-scale distributions of adjacent pixel sequence pairs: (a) horizontal of original image, (b) vertical of original image, (c) diagonal of original image, (d) horizontal of scrambled image using LNBT-1, (e) vertical of scrambled image using LNBT-1, (f) diagonal of scrambled image using LNBT-1, (g) horizontal of scrambled image using LNBT-2, (h) vertical of scrambled image using LNBT-2, (i) diagonal of scrambled image using LNBT-2, (j) horizontal of scrambled image using LNBT-3, (k) vertical of scrambled image using LNBT-3, and (l) diagonal of scrambled image using LNBT-3.

12(l) is obviously better. Therefore, it can be seen from Table 4 and Figure 12 that LNBT has good image scrambling performance.

5.3. Conclusions. This paper gives a new kind of scrambling transform, which is one-dimensional and is called LNBT with three parameters. LNBT is bijective, and thus LNBT is invertible and is a permutation on a finite subset of the natural numbers. Since LNBT has good scrambling performance, it can be used to shuffle two-dimensional image; in fact, it can be directly used to scrambled one-dimensional audio signal and also used as a tool of encryption.

Conflicts of Interest

These authors declare that there are no conflicts of interest regarding the publication of this paper.

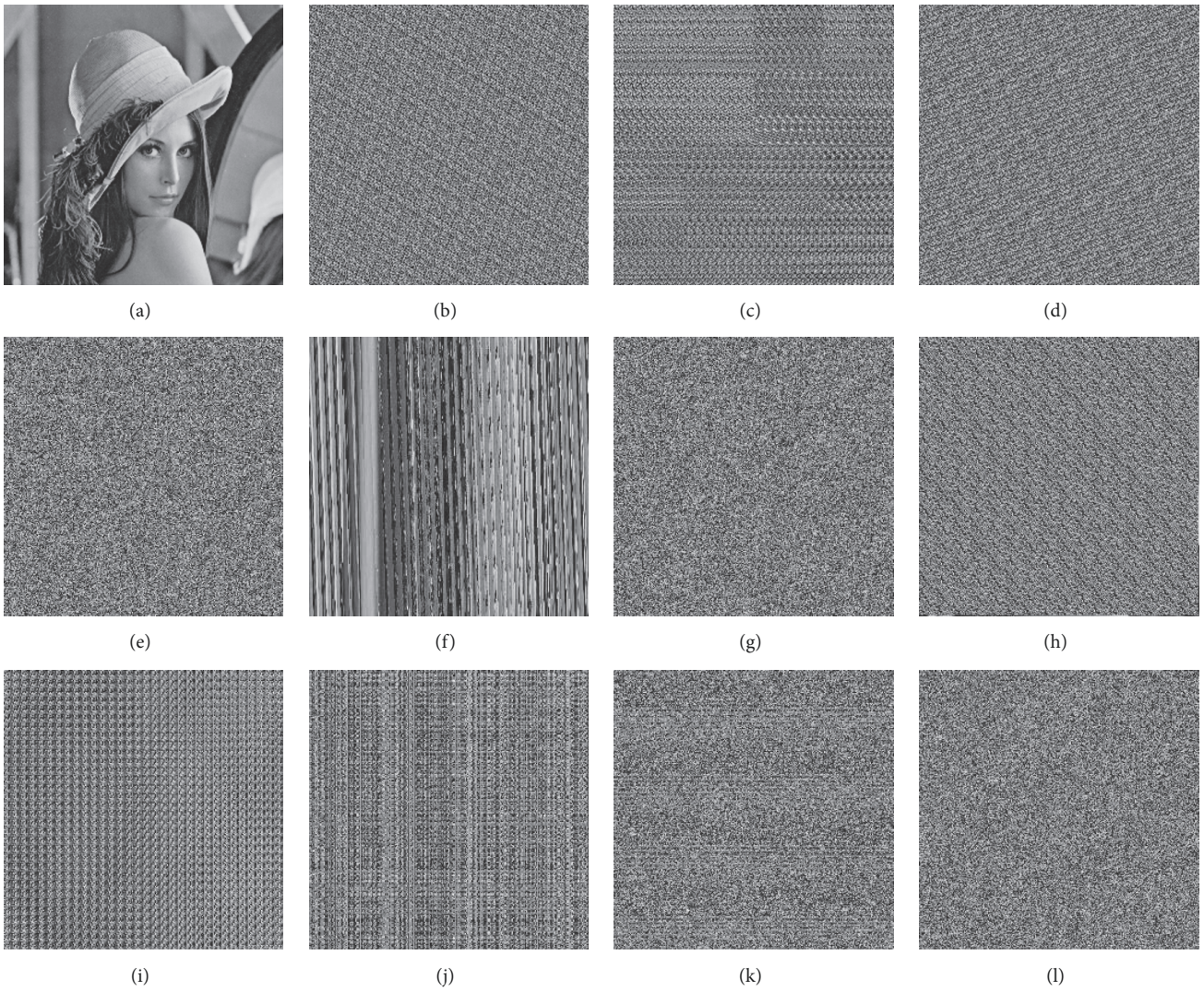


Figure 12: Scrambled images: (a) original image, (b) 6-round 2D Arnold transform, (c) 11-round discrete Baker map, (d) 13-round subaffine transform, (e) 13-round cellular automata transform, (f) 32-round Fibonacci transform, (g) 19-round Hilbert curve transform, (h) 27-round magic cube transform, (i) 3-round 2×2 sampling, (j) 29-round LNBT-1, (k) 32 LNBT-2, and (l) 24-round LNBT-3.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant 61170320, the Natural Science Foundation of Guangdong Province of China under Grant S2011040002981, and the Science and Technology Development Fund of Macau under Grant 048/2016/A2 and Grant 0012/2018/A1.

References

[1] N. A. Abbas, "Image encryption based on independent component analysis and arnold's cat map," *Egyptian Informatics Journal*, vol. 17, no. 1, pp. 139–146, 2016.

[2] G. Ye and K.-W. Wong, "An efficient chaotic image encryption algorithm based on a generalized Arnold map," *Nonlinear Dynamics*, vol. 69, no. 4, pp. 2079–2087, 2012.

[3] R. Ye, "A novel chaos-based image encryption scheme with an efficient permutation-diffusion mechanism," *Optics Communications*, vol. 284, no. 22, pp. 5290–5298, 2011.

[4] M. A. Gondal, Abdul Raheem, and I. Hussain, "A Scheme for Obtaining Secure S-Boxes Based on Chaotic Baker's Map," *3D Research*, vol. 5, no. 3, 2014.

[5] I. Hussain, T. Shah, M. A. Gondal, and H. Mahmood, "Efficient method for designing chaotic S-boxes based on generalized Baker's map and TDERC chaotic sequence," *Nonlinear Dynamics*, vol. 74, no. 1-2, pp. 271–275, 2013.

[6] L. Liu and S. Miao, "An image encryption algorithm based on Baker map with varying parameter," *Multimedia Tools & Applications*, pp. 1–17, 2016.

[7] M. Carli, F. Battisti, M. Cancellaro, G. Boato, and A. Neri, "Joint watermarking and encryption of color images in the fibonacci-haar domain," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, 2009.

[8] X. Ji, S. Bai, and G. Zhu, "Image encryption and compression based on the generalized knights tour, discrete cosine transform and chaotic maps," *Multimedia Tools & Applications*, pp. 1–15, 2016.

[9] K. Muhammad, M. Sajjad, I. Mehmood, S. Rho, and S. W. Baik, "A novel magic LSB substitution method (M-LSB-SM) using multi-level encryption and achromatic component of an image," *Multimedia Tools and Applications*, vol. 75, no. 22, pp. 14867–14893, 2016.

[10] D. Qi, J. Zou, and X. Han, "A new class of scrambling transformation and its application in the image information covering," *Science in China, Series E: Technological Sciences*, vol. 43, no. 3, pp. 304–312, 2000.

[11] S. Roy and A. K. Pal, "A robust blind hybrid image watermarking scheme in RDWT-DCT domain using Arnold scrambling," *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 3577–3616, 2017.

[12] L. Sun, J. Xu, and S. Liu, "A robust image watermarking scheme using Arnold transform and BP neural network," *Neural Computing & Applications*, pp. 1–16, 2017.

[13] J. Li, Q. Lin, C. Yu, X. Ren, and P. Li, "A QDCT- and SVD-based color image watermarking scheme using an optimized encrypted binary computer-generated hologram," *Soft Computing*, pp. 1–19, 2016.

[14] M.-J. Tsai, "Wavelet tree based digital image watermarking by adopting the chaotic system for security enhancement," *Multimedia Tools and Applications*, vol. 52, no. 2-3, pp. 347–367, 2011.

[15] T. D. Nguyen, S. Arch-int, and N. Arch-int, "A novel secure block data-hiding algorithm using cellular automata to enhance the performance of JPEG steganography," *Multimedia Tools and Applications*, vol. 74, no. 15, pp. 5661–5682, 2015.

[16] B. L. Gunjal and S. N. Mali, "MEO based secured, robust, high capacity and perceptual quality image watermarking in DWT-SVD domain," *SpringerPlus*, vol. 4, no. 1, pp. 1–16, 2015.

[17] J. J. Ranjani, "Data hiding using pseudo magic squares for embedding high payload in digital images," *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 3715–3729, 2017.

[18] L. Yang and K. Chen, "On the orders of transformation matrices (mod n) and two types of generalized Arnold transformation matrices," *Science China Information Sciences*, vol. 47, no. 5, pp. 655–668, 2004.

[19] N. Jiang, L. Wang, and W.-Y. Wu, "Quantum Hilbert Image Scrambling," *International Journal of Theoretical Physics*, vol. 53, no. 7, pp. 2463–2484, 2014.

[20] A. V. Diaconu, A. Costea, and M.-A. Costea, "Color image scrambling technique based on transposition of pixels between RGB channels using Knight's moving rules and digital chaotic map," *Mathematical Problems in Engineering*, vol. 2014, Article ID 932875, 15 pages, 2014.

[21] J. C. Zou, G. F. Li, and D. X. Qi, "Generalized Gray code and its application in the scrambling technology of digital images," *Applied Mathematics A Journal of Chinese Universities*, vol. 3, no. 3, pp. 363–370, 2002.

[22] R.-G. Zhou, Y.-J. Sun, and P. Fan, "Quantum image Gray-code and bit-plane scrambling," *Quantum Information Processing*, vol. 14, no. 5, pp. 1717–1734, 2015.

[23] A. L. Abu Dalhoum, A. Madain, and H. Hiary, "Digital image scrambling based on elementary cellular automata," *Multimedia Tools and Applications*, vol. 75, no. 24, pp. 17019–17034, 2016.

[24] A. L. A. Dalhoum, B. A. Mahafzah, A. A. Awwad, I. Aldhamari, A. Ortega, and M. Alfonseca, "Digital image scrambling using 2D cellular automata," *IEEE MultiMedia*, vol. 19, no. 4, pp. 28–36, 2012.

[25] R. Ye and H. Li, "A novel image scrambling and watermarking scheme based on cellular automata," in *Proceedings of the International Symposium on Electronic Commerce and Security, ISECS 2008*, pp. 938–941, chn, August 2008.

[26] F. Zhao X, "Digital image scrambling based on the baker's transformation," *Journal of Northwest Normal University*, 2003.

[27] S. Bai and Chongqing., "Property of sub-affine transformation and its application," *Journal of Computer Aided Design & Computer Graphics*, 2003.

[28] W. Ding, W. Yan, and D. Qi, "Digital image scrambling," *Progress in Natural Science*, vol. 11, no. 6, pp. 454–460, 2001.

[29] https://en.wikipedia.org/wiki/Negative_base.

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

[31] J.-L. Liu, "Efficient selective encryption for JPEG 2000 images using private initial table," *Pattern Recognition*, vol. 39, no. 8, pp. 1509–1517, 2006.

[32] A. Moumen, M. Bouye, and H. Sissaoui, "New secure partial encryption method for medical images using graph coloring problem," *Nonlinear Dynamics*, vol. 82, no. 3, pp. 1475–1482, 2015.