Image encryption using random sequence generated from generalized information domain

Xia-Yan Zhang(张夏衍)[1], Guo-Ji Zhang(张国基)[2,†], Xuan Li(李璇)[3], Ya-Zhou Ren(任亚洲)[4], and Jie-Hua Wu(伍杰华)[1]

[1] School of Computer Science and Engineering, South China University of Technology, Higher Education Mega Centre, Guangzhou 510006, China
[2] School of Mathematics, South China University of Technology, Guangzhou 510641, China
[3] School of Software, Fujian Normal University, Fuzhou 350007, China
[4] School of Computer Science and Engineering, Big Data Research Center, University of Electronic Science and Technology of China, Chengdu 611731, China

(Received 12 September 2015; revised manuscript received 18 December 2015; published online 31 March 2016)

A novel image encryption method based on the random sequence generated from the generalized information domain and permutation–diffusion architecture is proposed. The random sequence is generated by reconstruction from the generalized information file and discrete trajectory extraction from the data stream. The trajectory address sequence is used to generate a P-box to shuffle the plain image while random sequences are treated as keystreams. A new factor called drift factor is employed to accelerate and enhance the performance of the random sequence generator. An initial value is introduced to make the encryption method an approximately one-time pad. Experimental results show that the random sequences pass the NIST statistical test with a high ratio and extensive analysis demonstrates that the new encryption scheme has superior security.

Keywords: image encryption, random number generator, cryptography

PACS: 42.30.Va **DOI:** 10.1088/1674-1056/25/5/054201

1. Introduction

With the development of information technology and the Internet, digital images have been widely used in many areas. The security of images in some special areas, such as health systems, military systems, geographic information systems (GIS) or images of business and personal privacy has become an important issue. Image encryption is different from text encryption due to some inherent features, such as bulk data capacity, high correlation among pixels and high redundancy. Thus, most conventional ciphers, such as DES (data encryption standard), IDEA (international data encryption algorithm), AES (advanced encryption standard), etc. are not suitable for image encryption especially in real time encryption.

There is a wide range of algorithms used in image encryption such as cellular automaton (CA),[1–3] DNA computing,[4–6] and chaotic systems. Chaotic systems have several significant features favorable to secure communications, such as ergodicity, sensitivity to initial conditions, control parameters and random-like behavior, which can be connected with some conventional cryptographic properties of good ciphers, such as confusion and diffusion. After Matthews proposed the chaotic encryption algorithm in 1989,[7] increasingly research of image encryption technology is based on chaotic systems.

Many chaos-based cryptography schemes have shortcomings. One is the short cycle length resulting from the finite precision of computers which makes various attacks possible.[8,9] To extend the cycle length of chaotic systems, some chaotic stream ciphers utilized dual chaotic systems or hyper chaotic systems.[10–13] Another problem is that the key stream generated by the chaotic map is completely dependent on the secret key and remains unchanged to any plain image, which is insecure against chosen plain text attack.[14] In Ref. [14], a new block encryption scheme was proposed by combining circular bit shift and XOR operations based on iterating a chaotic map. Li *et al.*[15] showed that there are some security defects in such a scheme (proposed in Ref. [14]) and most elements in the underlying chaotic pseudorandom bit sequence can be obtained by a differential known plain text attack using only two known plain texts. To solve this problem, some approaches that try to make the encryption process or the key stream related to the plain image were proposed. In Ref. [16], a chosen plain text attack of Ref. [14] was presented because of the plain text-independent key stream, and an improvement was suggested by using a ciphertext feedback mechanism. Xu *et al.*[17] analyzed and improved the above algorithms by using Chen chaotic system instead of a logistic map for better random number sequence and setting the parameter of the Chen map using the last one byte of encrypted plain text after every iteration. Wang *et al.*[18] proposed an algorithm using a logistic map in which the shuffling and diffusion are performed simultaneously and the chaotic maps' states are altered according to the encrypted pixel. Zhang *et al.*[19] employed a plain text

†Corresponding author. E-mail: magjzh@scut.edu.cn

© 2016 Chinese Physical Society and IOP Publishing Ltd

feedback in permutation using the max and sum of the pixel value of the plain image and a plaintext/ciphertext feedback in diffusion. The permutation process behaves in a "one-time pad" manner and the diffusion process is sensitive to changes of plain image. Chen *et al.*[20] presented an image encryption scheme with a dynamic state variables selection mechanism. The state variables generated from chaotic systems are distributed to each pixel dynamically and pixel-relatively in both permutation and diffusion procedures. Dong[21] proposed a color image encryption scheme using one-time keys based on coupled chaotic systems. It employed secure hash algorithm 3 (SHA-3) combined with the initial keys to generate the new keys which would change in each encryption process.

The permutation–diffusion structure[22] is the most common and effective encryption architecture and has become the basis of many chaotic image encryption schemes since Fridrich developed a chaos-based image encryption scheme of this structure in 1998,[23] which was first cryptanalyzed in Ref. [24] using a chosen-ciphertext attack. It is composed of two steps: pixel permutation and diffusion. In the permutation process, the positions of image pixels are changed. In the diffusion process, the pixel values are modified sequentially so that a tiny change for one pixel can spread out to almost all pixels in the whole image. Zhang *et al.*[25] used a skew tent map to generate random sequences then sort the sequence to get index sequence as a P-box with the same size of the plain image which totally shuffled the plain image. Zhang *et al.*[26] sorted random chaotic sequences to get the index sequence as small permutations and then constructed large permutation from the small permutations and totally shuffled the plain image. General permutation-only image encryption algorithms have been quantitatively cryptanalyzed[27] and one round permutation–diffusion is not secure enough especially against chosen plain text attack.[28] Therefore, many image encryption algorithms employ the permutation–diffusion process by two rounds or more.

Zhang *et al.*[29] proposed a random number generator based on discrete trajectory transform in generalized information domain (RGDG) which has superior randomness. RGDG has a time-related initial value so that it can generate different random sequences every time. In this paper, our main contribution includes: 1) We introduce a new factor called drift factor (DF) and develop an improved algorithm of RGDG called RGDG-DF, which has an increased speed. 2) RGDG has an initial value IV $= \{t, r, u\}$ where t is system time, r is a system random number and u is user's personalized input. In RGDG-DF, IV = {ST, SI, UI}, where ST is system time, SI is system identifier that is unique, such as hard disk number of the machine and UI is user input. 3) IV is employed in RGDG-DF in a feedback mechanism. In every encryption process, IV is different so that random sequences generated by RGDG-DF are different by time and space. The image encryption scheme is an approximately "one-time pad". 4) We propose an image encryption scheme based on RGDG-DF. The trajectory address sequence generated by RGDG-DF is used to create a P-box with the same size of plain image. A new permutation method is proposed to shuffle the positions of pixels totally using the P-box. The random sequences of RGDG-DF are used as keystreams in the diffusion steps. 5) This algorithm has a generalized key which only needs to be settled in the first time at both communication sides. No secret key needs to be exchanged during the encryption process. Experimental results show that the random sequences pass the NIST statistical test[30] with a high ratio and the new encryption scheme has feasibility and superiority.

The rest of the paper is organized as follows. In Section 2, we describe the generalized information domain, RGDG, and the proposed RGDG-DF. In Section 3, we describe the scheme of image encryption and decryption. The experimental results and security analysis are given in Section 4. Finally, Section 5 concludes the paper.

2. Random sequence generated by discrete trajectory extraction with drift factor

2.1. Generalized information domain and rGDG

Generalized information domain (GID) is the space of all kinds of digital information stored in digital equipment or transferred over the Internet that can be expressed by binary code. With the development of the information technology and the Internet, GID is so large that it can be seen as an infinite space to some extent. In this paper, we denote the generalized information file as GIF which refers to the file in GID. Because of the diversity of users in cultural background, career, interests, habits and so on, the user's choice of GIF from GID can be seen as random behavior and an entropy source that is easy to get without any other facility. Zhang *et al.*[29] proposed a random number generator based on discrete trajectory transform in the generalized information domain (RGDG). In RGDG, they first reconstruct the GIF and user input using shift operation with restriction condition to get the data stream which has good statistical properties and appropriate length denoted by background (BG). Then, a discrete trajectory extraction (DTE) method is proposed to extract the random number from BG. A system random source called initial value IV $= \{t, r, u\}$ where t is system time, r is a system random number, and u is the user's personalized input that is introduced. IV is used to calculate the extraction position and the extracted data is used in updating the IV in a feedback mechanism. Experimental results showed that RGDG had superior randomness and passed the NIST statistical test with a high ratio. However, we found that RGDG spends most of the

time on calculating extraction positions. To speed up RGDG and preserve its good property at the same time, we propose an accelerated scheme based on drift factor (DF) in this paper, called RGDG-DF. DF is 32-bit vector used in the process of calculating extraction positions in Section 2.3.

2.2. Reconstruction from GIF

According to Ref. [29], GIFs usually have bad statistics properties such as low information entropy, high correlation coefficient, etc. To generate random number sequences, we also reconstruct GIF first to get a sequence with an appropriate length and good statistics properties which is called BG.

The reconstruction process includes n rounds. We note GIF as RB_0 (reconstruction block) and the result of the ith round of reconstruction as $\mathrm{RB}^i = \{\mathrm{RB}^i_j\}$ $(j = 0,1,2,\ldots,l_1 - 1)$, where RB^i_j is an 8-bit data. In each round, the reconstruction process can be presented as follows:

$$\mathrm{RB}^i = f(\mathrm{RB}^{i-1}, s_i, l_i),\ \ s_i \in \mathbb{N},\ \ l_i \in \mathbb{N}^+,\ \ (i = 1,2,\ldots,n). \quad (1)$$

In Eq. (1), parameter $(S,L) = \{(s_i,l_i)\}$ $(i = 1,2,\ldots,n)$, where s_i means the starting position of round i and l_i means the length of sequence of round i. The reconstruction process is given below in detail

$$\mathrm{RB}'_j = \mathrm{RL}\left(\mathrm{RB}^{i-1}_{(s_i+j) \bmod l_{i-1}}, \left\lfloor \frac{s_i + j}{l_{i-1}} \right\rfloor\right), \quad (i = 1,2,\ldots,n;\ j = 0,1,\ldots,l_{i-1}), \quad (2)$$

$$\mathrm{RB}^i_j = \begin{cases} \mathrm{RB}'_j, & c_{\mathrm{RB}'_j} - c_{\min} < q_1, \\ \min\{k | c_k = c_{\min}\}, & c_{\mathrm{RB}'_j} - c_{\min} \geq q_1, \end{cases} \quad (3)$$

where RL means rotate left shift, and $\mathrm{RL}(\mathrm{RB}^{i-1}_{(s_i+j) \bmod l_{i-1}}, \lfloor (s_i + j)/l_{i-1} \rfloor)$ means rotate left shift $\mathrm{RB}^{i-1}_{(s_i+j) \bmod l_{i-1}}$ by $\lfloor (s_i + j)/l_{i-1} \rfloor$ bits.

In Eq. (3), a counter vector $C = \{c_k\}$ $(k = 0,1,\ldots,255)$ is introduced, where c_k records the frequency of code k during the extraction process of each round and $c_{\min} = \min\{c_k\}$ $(k = 0,1,\ldots,15)$. At the beginning of each round, C will be reset to zero. The restriction condition $c_{\mathrm{RB}'_j} - c_{\min} < q_1$ is used to control the frequency distribution of each code of RB^i to be balanced. The restriction factor q_1 is used to control the degree of the balance level. If q_1 gets smaller, the restriction condition becomes more critical and the frequency distribution of the code sequence is more balanced. It is proved in Ref. [29], if $l_i > 255 \times q_1$, each code of RB^i exits and the frequency difference of any two codes is no more than q_1. After n rounds of reconstruction, we obtain a sequence RB^n and note it as $\mathrm{BG} = \{b_i\}$, where $b_i \in \{0,1,\ldots,255\}$ $(i = 0,1,2,\ldots,l_n - 1)$.

2.3. Random sequence extraction

BG cannot be used as random sequence directly because it has some regular pattern, not random. Therefore, an extraction method is used to generate random number sequences. The process of generating random number sequences SK and trajectory sequence PK from BG can be presented as $(\mathrm{SK},\mathrm{PK}) = f(\mathrm{BG},\mathrm{IV},D)$, where f is the extraction function. In each extraction operation, four bits of data are extracted from BG and added into the final random sequence based on some restriction condition. We introduced another counter vector denoted by $C' = \{c'_k\}$ $(k = 0,1,\ldots,15)$, where c_k records the frequency of code k during the extraction process, C' is initialized as zero and $c'_{\min} = \min\{c'_k\}$ $(k = 0,1,\ldots,15)$.

The extraction process is given as follows.

Step 1 Set $\mathrm{IV} = \{\mathrm{ST},\mathrm{SI},\mathrm{UI}\}$. ST is system time generated by the computer system and SI is system unique identifier such as hard disk number of the machine. UI depends on user's input, such as a number, a sentence, etc. The size of ST and SR is four bytes in standard configuration.

Step 2 Hash transformation of IV. Choose an irreversible, strong, collision-free hash function to transform IV to a vector $\mathrm{AI} = (x_0,x_1,\ldots,x_m)$. The size of x_i $(i = 0,1,\ldots,m)$ is 32 bits.

Step 3 Calculate the extraction position, trajectory address (ta) using

$$\mathrm{ta} = \left[\sum_{i=1}^{m}\left(a_i \prod_{j=0}^{i-1} d_j\right) + a_0\right] \bmod (8l_s), \quad (4)$$

where $a_i = x_i \bmod d_i$, $D = (d_0,d_1,\ldots,d_m)$, and $d_i \in \mathbb{Z}^+$ $(i = 0,1,\ldots,m)$.

Step 4 (Only in the first round) Initialize DF. Start at position ta (by bit) of BG and sequentially get 32 bits data noted ω to initialize DF using formula

$$\mathrm{DF} = \omega. \quad (5)$$

Step 5 Calculate ta using DF by formula

$$\mathrm{ta} = (\mathrm{ta} + \mathrm{DF}) \bmod (8l_s). \quad (6)$$

Step 6 Extract random number. Start at position ta (by bit) of BG and sequentially get four bits data noted M. If M cannot satisfy the restriction condition (i): $c'_M - c'_{\min} < q_2$, then let $\mathrm{ta} = \mathrm{ta} + 1$ and get four bits data sequentially again. Repeat this process until we get a data M satisfying condition (i).

Step 7 Update DF using formula

$$\mathrm{DF} = (\mathrm{DF} \ll 4)|M. \quad (7)$$

Step 8 Repeat Steps 5–7 four times and obtain M four times. Then go to Step 9 until we obtain sequence of requested length.

Step 9 Get new AI. $\mathrm{AI} = (x_0,x_1,\ldots,x_m)$ is seen as a $32 \times m$-bit-long sequence. Left shift four bit of AI and add four bits data M from Step 6 into the rightmost four bit of AI then get the new AI. Then go to Step 3.

Repeat Steps 3–9 until we get sequence SK of requested length. Another sequence SK$'$ can be generated similarly. In addition, we record the ta sequence denoted by PK which will

be used in the permutation part of the image encryption process.

In RGDG, analysis results show that the computation of Eq. (4) is time-consuming, taking about 60% of the whole time. This is because that equation (4) uses plenty of multiplications. To improve the efficiency of RGDG, in RGDG-DF, a 32-bit vector DF is introduced in Steps 4–7. In Step 4, DF is initialized by random data from BG and is different every time. Calculating ta using DF and the previous ta as in Step 6 mainly uses the addition operation which is more time-saving than Eq. (4). In Step 7, the extracted data M is used to update DF in a feedback mechanism which makes the extraction process more complex. Figure 1 shows the distribution of trajectory address (ta) and the changing process of DF sequence. We can see that ta has a balanced distribution in BG sequence and the changing process of DF is a random process like white noise.

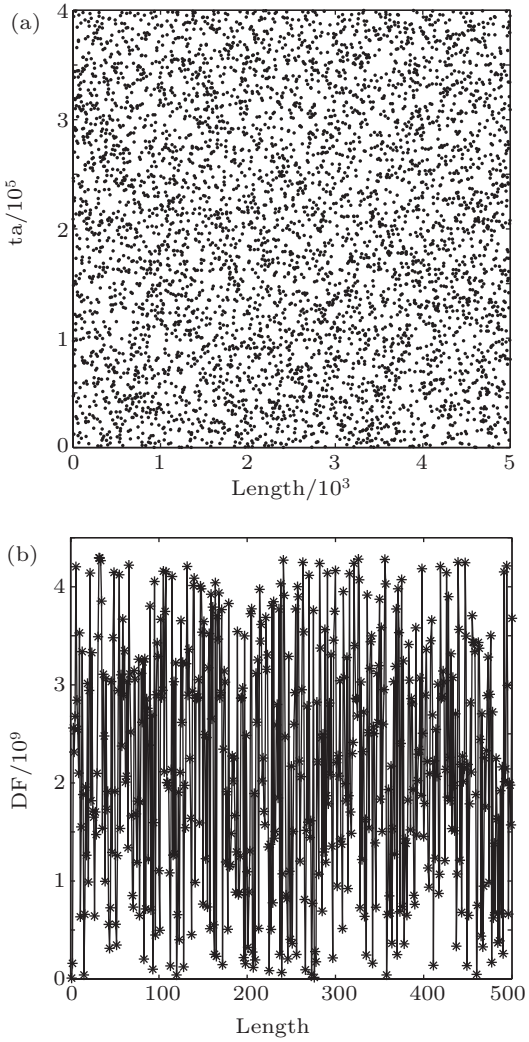


Fig. 1. (a) Distribution of ta sequence on BG (500000 bits), (b) changing process of DF.

2.4. Parameters

RGDG-DF has key $K = \{\text{GIF}, (S,L), D\}$, parameters q_1 and q_2. The suggested value ranges of parameters are summarized as below.

(I) GIF. GIF can be any type and any size, such as an image, a video, etc. Therefore, its space is infinite theoretically. In practice, we suggest the size of GIF between 0.5 KB and 10 MB for safety and space reasons.

(II) $(S,L) = \{(s_i, l_i)\}$ $(i = 1, 2, \ldots, n)$, where n is the reconstruction round. We test the influence of (S,L) under various GIFs with different length and content and find that when $n > 3$, the statistic properties of the reconstruction sequence are better. However, the efficiency decreases when n increases, so we suggest $n \in [3,7]$. From Section 2.2, we can see that the effective GIF which is actually used in the algorithm is related to (s_1, l_1), so we suggest $l_1 > 500$ in order to resist against brute force attack.

(III) $D = (d_0, d_1, \ldots, d_m)$. From Eq. (4), there is $a_i = x_i \bmod d_i$, which means that D determines the effective range of AI. Therefore, for long time usage, we suggest $\min\{d_i\} > 1000$.

(IV) q_1 and q_2. When the sequence length is less than 1000000 bits, tests show that when the restriction parameters q_1 and q_2 are less than 25, the restriction is too strong, the pass ratios of the NIST random tests[30] random excursions, random excursions variant and approximately entropy are less than 60%. When q_1 and q_2 are more than 70, the restriction is too weak, the pass ratios of the tests' random excursions, random excursions variant and approximate entropy are less than 85%. We suggest $25 \leq q_1 \leq 65$ and $25 \leq q_2 \leq 70$. When generating long sequences, we suggest increase q_1 dynamically such as adding q_1 by one every time when generating 1000000 bits.

3. The scheme of image encryption and decryption based on RGDG-DF

The encryption algorithm is based on the permutation–diffusion architecture. The random sequence PK is used to generate a P-box then shuffle the plain image while SK′ and SK are used to diffuse the shuffled image. Some papers[16,28] have shown that one round permutation–diffusion is not secure enough especially against chosen plain text attack. Therefore, we add a pre-diffusion to confuse the plain image. Moreover, a new shuffle method is proposed in the permutation process. When encrypting any two images the IVs are different in time and space because ST is different in time and SI is different in space. In practical situations, different IVs usually result in different keystreams. Thus, this encryption algorithm is an approximately one-time pad.

Without loss of generality, we assume the plain image is a 256 gray-scale image of size $M \times N$ which can be seen as a one-dimensional vector $P = (p_1, p_2, \ldots, p_{M\times N})$, where p_i denotes the gray level of the image pixel in the row floor (i/N) column mod (i,N). The algorithm can be described as below.

3.1. Encryption algorithm

Step 1 Generate a random byte c_0 and three random sequences PK, SK′, and SK as described in Section 2.

Step 2 Pre-diffusion. Compute the pixel value of the middle cipher image by the following formula:

$$c'_i = p_i \oplus c'_{i-1} \oplus \mathrm{sk}'_i, \quad (i = 1,2,\ldots,M\times N), \tag{8}$$

where $\oplus$ is bitwise XOR operator and c'_i is the gray level of the middle cipher image pixel.

Step 3 Permutation. (I) Sort $\mathrm{PK} = (\mathrm{pk}_1, \mathrm{pk}_2, \ldots, \mathrm{pk}_{M\times N})$ and obtain $\mathrm{PK}^* = \{\mathrm{pk}^*_1, \mathrm{pk}^*_2, \ldots, \mathrm{pk}^*_{M\times N}\}$.

(II) Find the positions of $\{\mathrm{pk}^*_1, \mathrm{pk}^*_2, \ldots, \mathrm{pk}^*_{M\times N}\}$ in PK and denote them as $T = \{t_1, t_2, \ldots, t_{M\times N}\}$, where $\mathrm{pk}^*_i = \mathrm{pk}_{t_i}$ $(i = 1,2,\ldots,M\times N)$.

(III) Use T as a P-box to shuffle $C' = \{c'_1, c'_2, \ldots, c'_{M\times N}\}$ and get $P' = \{p'_1, p'_2, \ldots, p'_{M\times N}\}$, where

$$\begin{cases} p'_{t_i} = c'_{t_{i+1}}, & (i = 1,2,\ldots,M\times N-1), \\ p'_{t_{M\times N}} = c'_{t_1}. \end{cases} \tag{9}$$

Step 4 Diffusion. Compute the pixel value of the cipher image by the following formula:

$$c_i = p'_i \oplus c_{i-1} \oplus \mathrm{sk}_i, \quad (i = 1,2,\ldots,M\times N), \tag{10}$$

where $\oplus$ is bitwise XOR operator and c_i is the gray level of the cipher image pixel.

3.2. Decryption algorithm

The decryption process is similar to the encryption process and is the inverse of the encryption process.

Step 1 Generate a random byte c_0 and three random sequences PK, SK′, and SK as described in Section 2.

Step 2 Generate P-box T using the same method as Step 2 in the encryption process.

Step 3 Get $P' = \{p'_1, p'_2, \ldots, p'_{M\times N}\}$ from the cipher image using the formula

$$p'_i = c_i \oplus c_{i-1} \oplus \mathrm{sk}_i, \quad (i = 1,2,3,\ldots,M\times N). \tag{11}$$

Step 4 Perform the reverse operation using P-box T and obtain the middle cipher image $C' = \{C'_1, C'_2, \ldots, C'_{M\times N}\}$ from $P' = \{p'_1, p'_2, \ldots, p'_{M\times N}\}$, where

$$\begin{cases} c'_{t_1} = p'_{t_{M\times N}}, \\ c'_{t_i} = p'_{t_{i-1}}, \quad i = 2,3,\ldots,M\times N. \end{cases} \tag{12}$$

Step 5 Obtain plain image $P = (p_1, p_2, \ldots, p_{M\times N})$ from $C' = \{C'_1, C'_2, \ldots, C'_{M\times N}\}$ by

$$p_i = c'_i \oplus c'_{i-1} \oplus \mathrm{sk}'_i, \quad (i = 1,2,\ldots,M\times N). \tag{13}$$

In our scheme, we propose a new shuffle method shown in Eq. (9) which has a simple inverse shown in Eq. (12). Equation (12) only uses the P-box T and does not need to calculate the inverse permutation sequence. This makes the decryption process more efficient.

4. Experimental results and performance analysis

4.1. Performance test and analysis of the random sequence

We choose 10 groups of different keys and parameters as shown in Tables 1 and 2 to generate 100 random sequences of 1000000 bits for each group and test them using NIST special publication 800-22,[30] a statistical package consisting of 15 tests for random and pseudo-random number generators for cryptographic applications. The results are shown in Table 3.

Table 1. Ten groups of keys and parameters for the NIST tests. Groups 1–5.

No.	1	2	3	4	5
GIF	IE11-Windows6.1.exe (1.98 MB)		Baboon.bmp(256 × 256) (257 KB)		
(q_1, q_2)	(26,26)	(30,70)	(35,35)	(40,60)	(45,45)
n	3	4	3	4	5
S	(1,2,3)×2400	(1,2,3,4) ×600	(1,2,3)×1000	(1,2,3,4) ×1200	(1,2,3,4,5) ×1400
L	(1,2,3)×44000	(1,2,3,4) ×30000	(1,2,3)×20000	(1,2,3,4) ×18000	(1,2,3,4,5) ×12000
D	(2151,4302, 6453)	(16291,30042, 43793,57544)	(1989,3978, 5967)	(3989,7978, 11967,15956)	(12021,24042, 36063,48084,60105)
AI	(41,18467, 6334)	(234,30031, 11744,4443)	(77,5628, 6232)	(68,6151, 18175, 22398)	(77,5628, 6232,29052,1558)

Table 2. Ten groups of keys and parameters for the NIST tests. Groups 6–10.

No.	6	7	8	9	10
GIF	fips1402annexa.pdf (89.8 KB)			ReadMe.htm (966 B)	
(q_1, q_2)	(50,50)	(55,55)	(60,40)	(65,65)	(65,70)
n	3	4	5	3	4
S	(1,2,3)×2000	(1,2,3,4)×44	(1,2,3,4,5) ×333	(1,2,3)×1800	(1,2,3,4) ×4200
L	(1,2,3)×40000	(1,2,3,4) ×44000	(1,2,3,4,5 ×36000)	(1,2,3)×90000	(1,2,3,4) ×100000
D	(6024,12048, 18072)	(6024,12048, 18072,24096)	(8691,14842, 20993,27144,33295)	(14894,28518, 42142)	(32299,34856, 37413,39970,42527)
AI	(90,15854, 12153)	(103,26079, 18073,24951)	(123,25034, 26954,5491,14890)	(228,8535, 8784)	(159,12194, 26852,8043,30046)

Table 3. Results of the NIST statistical test (at the 1% level).

Statistical test	Average pass ratio/%
Frequency	100.00
Block frequency	99.30
Cumulative sums (FORWARD)	100.00
Cumulative sums (REVERSE)	100.00
Runs	100.00
Longest run	99.70
Rank	99.00
FFT	98.80
Non overlapping template (template = 000000001)	99.80
Overlapping template	99.60
Universal	98.20
Approximate entropy	99.00
Random excursions ($x = -1$)	99.30
Random excursions variant ($x = -1$)	99.50
Serial	98.20
Linear complexity	99.50

From Tables 1–3, it can be seen that RGDG-DF has passed the test suit with a high ratio in different configurations. In some tests, i.e., the frequency test, cumulative sums (FORWARD), cumulative sums (REVERSE), and runs, the pass ratio is even 100%. This shows that RGDG-DF has superior randomness.

4.2. Key space analysis

Key space size is the total number of different keys which can be used in the encryption. A good encryption algorithm should be sensitive to the secret keys, and the key space should be large enough to make brute-force attack impossible.

In our algorithm, GIF together with parameters $(S,L),D$ denoted by $K = \{\text{GIF},(S,L),D\}$ can be seen as generalized key compared with traditional symmetric cryptography. This generalized key is different from the traditional key of a modern cryptography system such as DES, AES in three ways: (i) the length or size of K is not fixed, (ii) the key space is infinite, (iii) K can be stored in the facility of both sides of the communication and does not need to be transferred in a long time. GIF can be any digital file from the GID so its space is infinite theoretically. As mentioned before, we suggest that the size of GIF should range from 0.5 KB to 10 MB. Suppose the size of GIF is 1KB, then the space of GIF is $2^{1024\times8} = 2^{8192}$. $(S,L) = \{(s_i,l_i)\}$, $(i = 1,2,\ldots,n)$, where s_i and l_i are 32-bits integer and $n \in [3,7]$. Suppose $n = 3$, then the space of (S,L) is $2^{32\times2\times3} = 2^{192}$. $D = (d_0,d_1,\ldots,d_m)$, $(i = 0,1,\ldots,m)$, where d_i is 32-bits long. If $m = 4$, then the space of D is $2^{32\times5} = 2^{160}$. Thus, the key space of our algorithm is at least $2^{8192+192+160}$=2^{8544} which is so huge that it can prevent brute-force attack.

4.3. Statistical analysis

For experimental analysis, IV remains unchanged in Subsection 4.3–4.5 and the corresponding AI is given in Table 4. Key $K_0 = \{\text{GIF}_0,(S_0,L_0),D_0\}$ and parameters used in Subsections 4.3–4.5 are also shown in Table 4.

Table 4. Key K_0 and parameters.

GIF_0	S_0	L_0	D_0	AI_0	q_1	q_2
Baboon.bmp	(1,2,3,4) ×400	(1,2,3,4) ×60000	(4982,9895, 14808,19721)	(381,22190, 29200,5949)	47	47

4.3.1. Histogram analysis

Image histogram is a very important feature in image analysis. Figures 2(a)–2(d) depict the plain images, cipher images, and their histograms, respectively. From these figures, we can see that the histogram of the cipher images is fairly uniform and is significantly different from that of the plain image.

4.3.2. Correlation analysis

Correlation between pixels is an important intrinsic feature of an image. To resist statistical analysis, a good encryption algorithm should remove the high correlation of the plain image. We select all the pairs of two-adjacent pixels in vertical, horizontal, and diagonal directions from the plain image and the cipher image, and calculate the coefficient by

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{14}$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N} [x_i - E(x)]^2, \tag{15}$$

$$\text{cov}(x,y) = \frac{1}{N}\sum_{i=1}^{N} [x_i - E(x)][y_i - E(y)], \tag{16}$$

$$r_{xy} = \frac{\text{cov}(x,y)}{\sqrt{D(x)D(y)}}, \tag{17}$$

where x and y are gray-scale values of two adjacent pixels in the image and N is the total number of pixels.

We test the correlation coefficients of adjacent pixels in the plain image "Lenna" and its cipher image and the result is listed in Table 5. It shows that the correlation coefficients of adjacent pixels in the cipher image are negligible around zero. Moreover, we choose 1000 pairs of two horizontally adjacent pixels and their correlation is shown in Fig. 3. These results substantiate that the proposed algorithm has removed the strong correlation among neighboring pixels of the plain image.

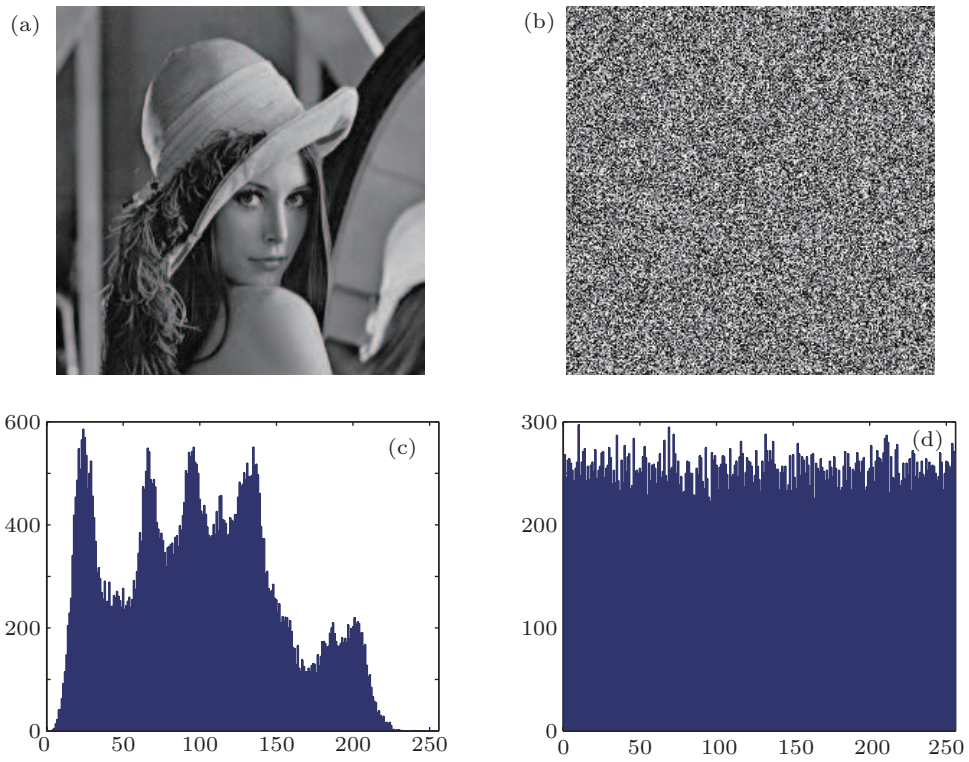


Fig. 2. (color online) (a) Plain image, (b) cipher image, (c) histogram of the plain image, and (d) histogram of the cipher image.

Table 5. Correlation coefficients of two adjacent pixels in the plain image and cipher image.

Direction	Plain image	Our algorithm	Ref. [18]	Ref. [19]	Ref. [25]
Horizontal	0.939918	0.000369102	0.0014	0.0035	–0.000848277
Vertical	0.969235	0.000636604	0.0171	0.00247	0.00370914
Diagonal	0.937176	0.000996246	0.0054	0.00107	–0.000188985

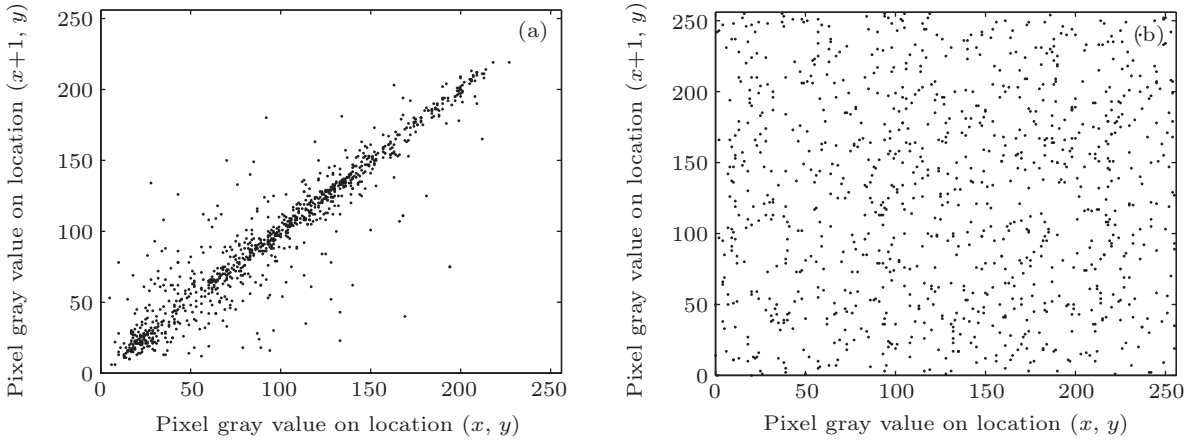


Fig. 3. Correlations of two horizontally adjacent pixels: (a) correlation of the plain image and (b) correlation of the cipher image.

4.3.3. Information entropy analysis

The information entropy is an important feature of the randomness and the information entropy $H(s)$ of a message source s can be calculated as

$$H(s) = \sum_{i=0}^{2^N-1} p(s_i)\log_2 \frac{1}{p(s_i)}, \tag{18}$$

where $p(s_i)$ denotes the probability of symbol s_i. For a true random 256 gray-scale image, the entropy should be 8. The entropy of cipher images of three plain images "Lenna", "Baboon", and "Pepper" is shown in Table 6. The entropy values of the cipher images are very close to the theoretical value eight which means the information leakage in the encryption process is negligible. The results of the proposed algorithm

and three existing algorithms[18,19,25] are shown in Table 7.

Table 6. Entropy value for the cipher images.

Plain image	Entropy value
Lenna	7.99793
Baboon	7.99929
Pepper	7.99922

Table 7. Entropy value for cipher images of "Lenna".

Cipher image	Entropy value
Ref. [18]	7.9974
Ref. [19]	7.9973
Ref. [25]	7.99748
Proposed	7.99793

4.4. Sensitivity analysis

In order to measure the different ranges between two images, we use NPCR (number of pixels change range) and UACI (unified average changing intensity),

$$\mathrm{NPCR} = \frac{\sum_{i,j} D(i,j)}{W \times H} \times 100\%, \tag{19}$$

$$\mathrm{UACI} = \frac{1}{W \times H}\left[\sum_{i,j} \frac{|c_1(i,j) - c_2(i,j)|}{255}\right] \times 100\%, \tag{20}$$

where c_1 and c_2 are two images with the same size $W \times H$. If $c_1(i,j) = c_2(i,j)$, $D(i,j)$=1; otherwise $D(i,j) = 0$.

4.4.1. Key sensitivity analysis

Key sensitivity is an essential feature for any good cryptosystem which guarantees the security of the cryptosystem against the brute-force attack to some extent. In the proposed algorithm, the generalized key $K = \{\mathrm{GIF}, (S,L), D\}$ is composed of several parts so we run several tests for each part of the key K separately. We randomly change one bit of GIF 1000 times to get 1000 slightly different GIFs and keep (S,L), D unchanged. We encrypt image "Lenna" using K_0 and these 1000 keys and calculate the average NPCR and UACI of the cipher images using Eqs. (19) and (20). There are 29 results that NPCR = 0, UACI = 0. This is because the effective GIF is related to (s_1, l_1) and the only changed bit may not belong to the effective GIF. However, the effective GIF is large enough to resist the brute force attack. The average results of the 971 groups are shown in Table 8. Similarly, change one value of (S_0, L_0) by adding or minusing 1 separately and get a series keys. We encrypt image "Lenna" using K_0 and these keys and calculate the average NPCR and UACI of the cipher images. The average results are shown in Table 8. IV = {ST, SI, UI} sent with the cipher image is not part of the key, but is the unique variable of the encryption algorithm and when the key is fixed, IV is the main factor of the approximately one-time pad manner. Therefore, a tiny change in the IV should cause a substantial change in the cipher image. $\mathrm{AI} = (x_0, x_1, \ldots, x_m)$ is the hash transformation result of IV. For simplification, we test the sensitivity of AI instead of IV. We choose $\mathrm{AI}_0 = (x_0, x_1, \ldots, x_m)$ and K_0 to encrypt plain image "Lenna" and then encrypt the same image with the same K_0 and other slightly different AIs. The NPCR and UACI between the cipher images are given in Table 8.

Table 8. NPCR and UACI between cipher images with key K_0 and slightly different keys.

K	NPCR/%	UACI/%
GIF	99.6086	33.50182
(S,L)	99.60309	33.50534
D	99.62176	33.49288
AI	99.60233	33.5227

4.4.2. Plaintext sensitivity

In order to resist differential attack, a tiny change in the plain image should cause a substantial change in the cipher image. Given a 256 gray-scale plain image P of size 256×256 and get a P' which only has a single pixel difference in a random position (i,j), where $i, j = 0, 1, \ldots, 255$,

$$P'(i,j) = \begin{cases} P(i,j)+1, & \text{if } p(i,j) < 255, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

Encrypting P and P' obtains the cipher images C and C', then we calculate NPCR and UACI using Eqs. (19) and (20). We test 3 groups of plain images, randomly repeat this process 100 times and get the average NPCR and UACI shown in Table 9, respectively, which are very close to their expectation.

Table 9. NPCR and UACI between cipher images with slightly different plain images.

Plain image	NPCR/%	UACI/%
Lenna	99.5737	33.4789
Baboon	99.60791	33.44112
Pepper	99.61115	33.46517

4.5. Speed

We have used Microsoft Visual Studio 2010 to run RGDG,[29] RGDG-DF, encryption and decryption programs in a personal computer with an Intel Core i3 CPU 2.53 GHz, 2 GB memory and 250 GB hard-disk capacity, and the operation system is Microsoft Windows 7. The average speed of RGDG-DF is 8.29 M/s and the average speed of RGDG is 5.78 M/s. The average time of encryption/decryption on 256 gray-scale images of size 512 × 512 is 150 ms. Considering the speed of the Internet transmission, our algorithm can well satisfy the practical situation.

4.6. Resistance to known plain text and chosen plain text attacks and one-time pad

Compared with that in Ref. [29], we add a random value SI in the initial value IV. It can be deduced that when encrypting any two images, IVs must be different. ST is system time generated by computer system so when encrypting images at different times the STs are different. SI is system unique identifier such as hard disk number, mainboard number, etc. When encrypting the two images at the same time then they must be at different machines so that SIs are different. It is easy to notice that Eq. (4) is an injection of $A = (a_0, a_1, \ldots, a_m)$. Therefore, in a practical situation, different IVs usually result in different random streams. Thus, every encryption uses different keystreams and this encryption method is an approximately one-time pad. Therefore, the attacks proposed in Refs. [15], [16], and [28] become ineffective to this new scheme. The proposed scheme can easily resist the known plain text and the chosen plain text attacks.

4.7. Key management

Key management includes key storage, key backup/recovery, and key period. In our scheme, the key $K = \{\text{GIF}, (S, L), D\}$ can be seen as a generalized key compared with traditional symmetric cryptography. The process of the communication in practical usage is shown in Fig. 4. The key K only needs to be settled by both sides of the communication in the first time and can be stored at the machine of both sides. No secret key needs to be exchanged during the encryption process. IV and the cipher image are sent to the receiver on a public channel. Therefore, our scheme can tremendously reduce the risk of key leakage. For example, the sender and receiver choose the newest installation package of iTunes as GIF. The key $\{(S, L), D\}$ is exchanged on a safety channel. GIF can pretend to be an ordinary file in the computer and does not need to be stored securely. In addition, the backup GIF can be found on the Internet anytime which is very convenient. The period of ST is $2^{32}/(60 \times 60 \times 24 \times 365) = 136$ year (when the time unit is seconds). When a longer period is needed, a larger ST such as 64 bits could be applied, so in a quite long time, both the sender and receiver do not need to change the key for safety which saves sources from key distribution and exchange.

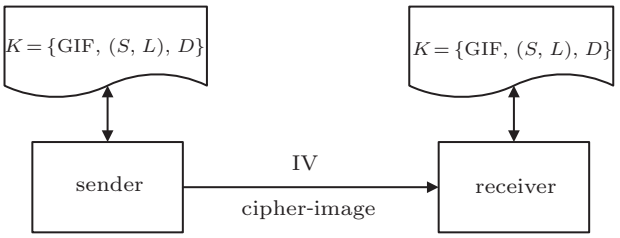


Fig. 4. Communication of the sender and receiver.

5. Conclusion

In this paper, we propose a novel image encryption method based on random sequence generated from generalized information domain and permutation–diffusion architecture is proposed. This algorithm takes user's random choice of the GIF as the entropy source and generates a random sequence by reconstruction from GIF and discrete trajectory extraction from the data stream. A new drift factor is introduced to accelerate and enhance the random generator. The trajectory address sequence is used to generate a P-box and then shuffle the plain image conpletely using a new permutation method and the random sequence is used as the keystream. A unique IV composed of ST, SI, and UI is introduced to make the encryption method an approximately one-time pad. Thus, the proposed scheme shows good resistance to the known plain text and the chosen plain text attacks. Experimental results show that the random sequences can pass the NIST statistical test with a high ratio. The key space is large enough to resist brute-force attacks. Statistical analysis shows that the cipher image of the scheme has high information entropy and low correlation coefficients so that it can resist statistical attack. The scheme has high key sensitivity and plain text sensitivity to anti differential attack. Moreover, the scheme has a high encryption speed. This algorithm has great advantages in practical use especially in some special areas which require high security such as military systems.

References

[1] Wang X and Luan D 2013 *Commun. Nonlinear Sci. Numer. Simulat.* **18** 3075
[2] Ping P, Xu F and Wang Z J 2014 *Signal Process.* **105** 419
[3] Abdo A A, Lian S, Ismail I A and Diab H 2013 *Commun. Nonlinear Sci. Numer. Simulat.* **18** 136
[4] Wei X, Guo L, Zhang Q, Zhang J and Lian S 2012 *J. Sys. Software* **85** 290
[5] Liu H and Wang X 2012 *Appl. Soft Comput.* **12** 1457
[6] Huang X and Ye G 2014 *Multimedia Tools Appl.* **72** 57
[7] Matthews R 1989 *Cryptologia* **13** 29
[8] Xiao D, Liao X and Wei P 2009 *Chaos, Solitons & Fractals* **40** 2191
[9] Li C, Li S, Chen G and Halang W A 2009 *Image Vision Comput.* **27** 1035
[10] Mirzaei O, Yaghoobi M and Irani H 2012 *Nonlinear Dyn.* **67** 557
[11] Zhu C 2012 *Opt. Commun.* **285** 29
[12] Wang Z, Huang X, Li Y X and Song X N 2013 *Chin. Phys. B* **22** 010504
[13] Luo Y L and Du M H 2013 *Chin. Phys. B* **22** 80503
[14] Xiang T, Liao X, Tang G, Chen Y and Wong K W 2006 *Phys. Lett. A* **349** 109
[15] Li C, Li S J, Alvarez G, Chen G and Lo K T 2007 *Phys. Lett. A* **369** 23
[16] Wang Y, Liao X, Xiang T, Wong K W and Yang D 2007 *Phys. Lett. A* **363** 277
[17] Xu S J, Chen X B, Zhang R, Yang Y X and Guo Y C 2012 *Phys. Lett. A* **376** 1003
[18] Wang X Y and Wang Q 2014 *Chin. Phys. B* **23** 030503
[19] Zhang L Y, Hu X B, Liu Y S, Wong K W and Gan J 2014 *Commun. Nonlinear Sci. Numer. Simulat.* **19** 3653
[20] Chen J, Zhu Z, Fu C, Yu H and Zhang L B 2015 *Commun. Nonlinear Sci. Numer. Simulat.* **20** 846

[21] Dong C 2014 *Signal Process.: Image Commun.* **29** 628
[22] Shannon C E 1949 *Bell System Technical J.* **28** 656
[23] Fridrich J 1998 *Int. J. Bifurc. chaos* **8** 1259
[24] Solak E, Cokal C and Yildiz O T 2010 *Int. J. Bifurc. chaos* **20** 1405
[25] Zhang G J and Liu Q 2011 *Opt. Commun.* **284** 2775
[26] Zhang X and Zhao Z 2014 *Nonlinear Dyn.* **75** 319
[27] Li C Q 2016 *Signal Process.* **118** 203
[28] Wang X and He G 2011 *Opt. Commun.* **284** 5804
[29] Zhang G J, Li X, Liu Q and Zhang X Y 2012 *Acta Phys. Sin.* **61** 485 (in Chinese)
[30] Rukhin A, Soto J, Nechvatal J, *et al*. 2010 *NIST Special Publication* 800-22*rev*1*a*