Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 6, June 2019, pp. 1692-1701
DOI: 10.23940/ijpe.19.06.p20.16921701

Improved Security for Android System based on Multi-Chaotic Maps using a Novel Image Encryption Algorithm

Ge Jiao[a,b,*], Lang Li[a,b], and Yi Zou[a,b]

[a]*College of Computer Science and Technology, Hengyang Normal University, Hengyang, 421002, China*
[b]*Hunan Provincial Key Laboratory of Intelligent Information Processing and Application, Hengyang, 421002, China*

Abstract

Aimed at the unsatisfactory effect of security in mobile cloud computing, we propose a novel image encryption algorithm by using the Improved Fruit Fly Optimization Algorithm -- Artificial Bee Colony (hereafter referred to as IFOA-ABC). First, the feature of task scheduling function in mobile cloud computing is described to set up the task scheduling function; second, the population of fruit fly optimization algorithm is initialized by using orthogonal array and quantitative techniques, and the boundary of fruit fly optimization algorithm is processed, the step size in search is adjusted dynamically, and the artificial bee colony algorithm is used to help fruit fly optimization algorithm develop the global optimal solution; in the simulation experiment, the IFOA-ABC algorithm has obvious advantages over other intelligent algorithms in the comparison of four indicators of mobile cloud task scheduling, which means that the proposed algorithm can effectively improve the privacy and efficiency of cloud computing scheduling.

Keywords: IFOA; ABC; mobile cloud computing; system security; algorithm improvement; task scheduling

(Submitted on March 20, 2019; Revised on April 12, 2019; Accepted on June 16, 2019)

1. Introduction

Market research firm Gartner released their third quarter 2018 Smartphone Market Report [1], and in the report, it showed that Android OS has reached 88.1% of market share. The privacy, security and reliability of the image information carried by it are increasingly valued, so it needs to be encrypted. Because image data has a large amount of information and high redundancy, traditional encryption algorithms such as AES and DES are used to encrypt and decrypt images, which has low efficiency and poor security.

Pak C et al. [2] designed a simple and efficient chaotic system by using the difference of two identical one-dimensional chaotic mapping output sequences and applied the system to image encryption to generate better chaos performance and better resist various attacks. Rostami M J et al. [3] proposed a chaotic window parallel image encryption algorithm based on Logistic mapping and used it for grayscale image encryption. The main idea is to decompose the image into blocks and encrypt it with XOR operation and chaotic window. The encryption algorithm with a larger key space, uniform histograms, NPCR and UACI metrics indicate that they can effectively defend against differential attacks.

Patidar V et al. [4] designed an improved substitution-diffusion image encryption algorithm based on chaotic standard and Logistic mapping, which realized fast image scrambling and diffusion and can effectively defend against plaintext attacks. However, these methods require a large amount of computation in the process of grayscale transformation or constructing a random sequence. If the methods are directly used in mobile device encryption, the speed of image encryption will be reduced. Tayde et al. [5] used the AES algorithm to encrypt files (including image files) on Android mobile phones, but did not analyze the efficiency of the algorithm. Tu [6] converted the bit plane of the image into a DNA plane, and the image encryption on the Android mobile phone is achieved through addition, subtraction, and XOR operations of the DNA sequence. Zhao [7] used Logistic and Chebyshev mapping to generate chaotic sequences to diffuse image pixels, which reduces the time and space complexity of the algorithm. A mobile phone image encryption software

* Corresponding author.
E-mail address: jiaoge@126.com

based on Android is developed, but pixel scrambling is omitted. Jiang [8] uses Baker's mapping to perform pixel scrambling and pixel diffusion through four-dimensional Cat mapping to complete Android platform image encryption, but there was no analysis of encryption time.

Existing image encryption algorithms are implemented more for the PC platform, and there are fewer encryption algorithms for the Android platform. This paper presents a hybrid chaotic image encryption algorithm based on the Android platform, and the histogram, key space, key sensitivity, pixel correlation, differential attack, information entropy attack, encryption/decryption time, etc. are analyzed and compared in detail. It has been proven that the efficiency of image encryption and decryption are greatly improved under the premise of ensuring the security of this algorithm.

2. Image Encryptions in Mobile Cloud Computing

The common method of chaotic system in image encryption is to use the initial value or control parameter of the chaotic system as the key of the encryption system. Then, the chaotic sequence is used to encrypt the image through multiple iterations of the chaotic map. Thus, the original image and the key are mixed and diffused. Common chaotic maps include Logistic map, Chebyshev map, and so on. Compared with the traditional image encryption technology, the image encryption technology based on chaos has advantages such as large key space, fast encryption speed, and easy implementation.

2.1. Logistic Map

Logistic map is a typical non-linear chaotic equation, which can generate complex chaotic behavior. The generated chaotic sequence has better randomness. The formula is defined as shown in Equation (1):

$$x_{n+1} = \mu x_n (1 - x_n),\ x_n \in (0,1) \tag{1}$$

Among them, the x_n is the n^{th} iteration quantity of Logistic chaotic mapping. When the system parameter $\mu \in (3.5699456, 4]$, the input and output of the Logistic map are all distributed on (0,1). The logistic map is in the chaotic state [9-11].

2.2. ChebyShev Map

ChebyShev map [11-12] has good initial sensitivity and long-term unpredictability of chaotic sequences. At the same time, as the number of iterations increases, the chaotic trajectories will be uniformly mixed, and the initial neighboring points will be separated by exponents, so it is suitable for the key control to generate chaotic sequences. As defined by the equation shown in Formula (2):

$$x_n + 1 = \cos(k \arccos x_n),\ x_n \in [-1,1] \tag{2}$$

Among them, k is the control parameter, when $k \geq 2$, $x_n \in [-1,1]$, the ChebyShev map is in chaos, and the initial value of the system iteration is determined by the key input by the user.

2.3. Improved Cat Map

Cat map [13] can only scramble $N \times N$ images. If you need to scramble $M \times N$ images, you need to improve them. The improved formula is shown in Equations (3)-(5).

$$\begin{bmatrix} x_i' \\ y_j' \end{bmatrix} = \begin{bmatrix} 1 & b \\ a & ab+1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \bmod(M, N) \tag{3}$$

$$x_i' = (x_i + a y_j) \bmod M \tag{4}$$

$$y_j' = (a x_i + ab y_j + a y_j) \bmod N \tag{5}$$

Assuming that the image $I_{M \times N}$ is scrambled to be $I'_{M \times N}$, and (x_i, x_j), (x_i', y_j') respectively represent the positions

of the pixels before and after the scrambling. $i \in [1, M]$, $j \in [1, N]$, a, b are positive integers.

Using Equations (6)-(8), we can calculate the degree of scrambling and the average of the image after each Cat map for the matrix of degree n, and the get the best number of iterations to make the image fully scrambled according to the scrambling degree and the mean value [14].

$$d(x_i, y_j) = \sqrt{(x_i - x_i')^2 + (y_j - y_j')^2} \tag{6}$$

$$E(d(x_i, y_j)) = \frac{1}{M \times N} \sum_{x=1}^{M} \sum_{y=1}^{N} d(x_i, y_j) \tag{7}$$

$$SH_n = \frac{1}{n} \sum_{k=1}^{n} a_k \frac{E(d_k(x_i, y_j))}{Var(d_k(x_i, y_j))} \tag{8}$$

$d(x_i, y_j)$ represents the pixel moving distance value, $E(d(x_i, y_j))$ represents the average value of the moving distance of the image, SH_n represents the n-order scrambling degree, and a_k represents the weighting coefficient, which embody the effect of different order distances in the degree of disorder.

3. Multi-Chaotic Chaotic Image Encryption Algorithm

3.1. Algorithm Design Idea

This paper designs a hybrid chaotic image encryption method. The algorithm flow chart is shown in Figure 1. The algorithm first uses the improved Cat map to fully scramble the pixel positions of the image according to the optimal number of iterations, and then randomly generates the parameters of the Logistic and ChebyShev iterations and the required initial key, respectively. After iterations to eliminate the transients, ChebyShev and Logistic were iterated again 3 times. The results of the iterations were used as the initial keys for encryption, thereby actually increasing the key space. Then, the pixels in the picture are taken. If the value of the pixel position is odd, use the ChebyShev iteration to obtain the encryption key. Otherwise, use the Logistic iteration to obtain the encryption key, thereby reducing the correlation between adjacent pixels. Then, use the three channels of R (red), G (green), and B (blue) of this pixel to perform exclusive-OR operation with B', R', and G' of the previously encrypted pixel, respectively.

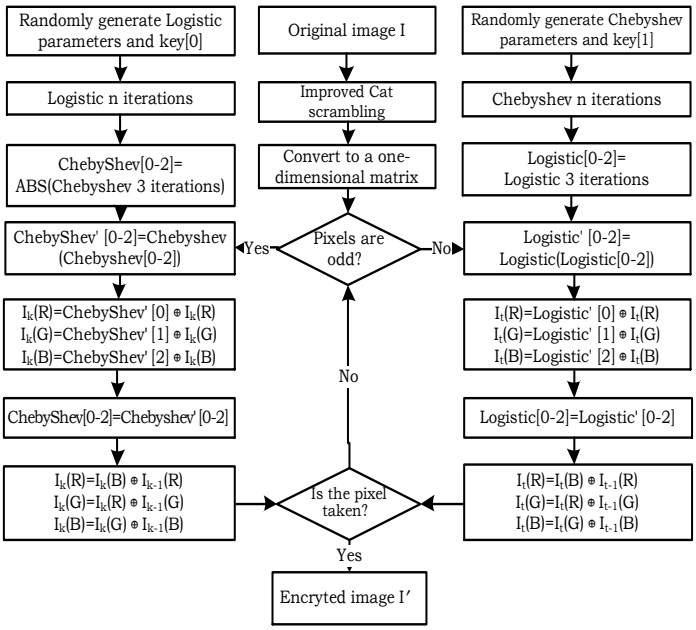


Figure1. Algorithm flow chart

3.2. Algorithm Implementation

The algorithm implementation includes the following steps：

Step 1 Take a color image *I* with an image size of $M \times N$;

Step 2 Cat scrambles for *I*;

Step 3 The two-dimensional matrix of pixel values of the image I is converted into a one-dimensional matrix $I[0, M \times N - 1]$;

Step 4 Generate two random keys as key[0] and key[1], key[0]∈(0,1), key[1]∈[-1,1], where key[0] is used as the initial key of Logistic iteration and key[1] is the initial key of the ChebyShev iteration;

Step 5 Using key[0] as the initial key, iterate *n* times (eliminating the impact of transients, 100 times in subsequent experiments) with Logistic map, and then iterate 3 times with ChebyShev. The results of the three ChebyShev iterations are taken as absolute values and stored in ChebyShev[0], ChebyShev[1], and ChebyShev[2], respectively;

Step 6 Using key[1] as the initial key, iterate *n* times with ChebyShev map and iterate it 3 times with Logistic. The absolute values of these three logistic iterations are stored in Logistic[0], Logistic[1], and Logistic[2], respectively;

Step 7 Take the pixels from *I* in the order of the pixels, and save the values when the pixels are not decrypted. If the current pixel position is an odd number (indicated by I_k), use the ChebyShev chaotic map to iterate with ChebyShev[0-2] generated in step (5) as the initial key. Record each result of iteration and save them to ChebyShev'[0-2]. Then, the three channels of the pixel (represented by *R*, *G*, *B*) are XORed by ChebyShev'[0-2], which are:

$$I_k(R) = \text{ChebyShev'}[0] \oplus I_k(R)$$
$$I_k(G) = \text{ChebyShev'}[1] \oplus I_k(G)$$
$$I_k(B) = \text{ChebyShev'}[2] \oplus I_k(B)$$

Then, assign ChebyShev[0-2] = ChebyShev'[0-2] for the next pixel encryption;

Step 8 If this pixel position is even (indicated with I_t), use Logistic chaotic map to iterate with Logistic[0-2] generated in step (6) as the initial key, and record and save the result to Logistic'[0-2] in each iteration. Then, the three channels of pixels are Exclusive OR Logistic' [0-2]), which are:

$$I_t(R) = \text{Logistic' [0]} \oplus I_t(R)$$
$$I_t(G) = \text{Logistic' [1]} \oplus I_t(G)$$
$$I_t(B) = \text{Logistic' [2]} \oplus I_t(B)$$

Then, assign Logistic' [0-2] to Logistic[0-2] as the next pixel encryption;

Step 9 Diffuse the encrypted pixels and take the components $I_{i-1}(R)$, $I_{i-1}(G)$, $I_{i-1}(B)$ of the pixel at the previous position of the pixel being encrypted to execute XOR with the components $I_i(B)$, $I_i(R)$, $I_i(G)$ of the pixel being encrypted:

$$I_i(R) = I_i(B) \oplus I_{i-1}(R)$$
$$I_i(G) = I_i(R) \oplus I_{i-1}(G)$$
$$I_i(B) = I_i(G) \oplus I_{i-1}(B)$$

Step10 Steps (7)-(9) are repeated until all the pixels in *I* are traversed and the ciphertext image I' is finally output.

3.3. Algorithm Description

The mixed chaotic encryption algorithm is described in C++ language as follows.

Multi-Chaotic Maps Image Encryption Algorithm

Input：Original Image I
Output： Ciphertext image IC

```
1:n=catBestIterations ();
2:IC=catMap(I,n);
3:Key[0]=randomKey(); Key[1]=randomKey();
4:U=randomRef(); K=randomRef()；
5:for(int i=0;i<=n;i++)
6:{
7:   Logistic=U*Key[0]*(1-Key[0]);
8:   Key[0]=Logistic;
9:   ChebyShev=cos(K*arcos(Key[1]));
10:   Key[1]=ChebyShev；
11:}
12:for(int i=0;i<=2;i++)
13:{
14:   ChebyShev[i]=fabs(cos(K*arcos(Key[0])));
15:   Logistic[i]=fabs(U*Key[1]*(1-Key[1]));
16:}
17:for(int j=1;j<=M*N-1;j++)
18:   if(j%2==1)
19:   {
20:      for(int i=0;i<=2;i++)
21:      {
22:         ChebyShev[i]=ChebyShev[i]*K*(1-ChebyShev[i]);
23:      }
24:      ICj(R)=ChebyShev[0]^ICj(R);
25:      ICj(G)=ChebyShev[1]^ICj(G);
26:      ICj(B)=ChebyShev[2]^ICj(B);
27:      ICj(R)=ICj(B)^ICj-1(R);
28:      ICj(G)=ICj(R)^ICj-1(G);
29:      ICj(B)=ICj(G)^ICj-1(B);
30:   }
31:   else
32:   {
33:      for(int i=0;i<=2;i++)
34:      {
35:         Logistic[i]=Logistic[i]*U*(1-Logistic[i]);
36:      }
37:      ICj(R)=Logisjic[0]^ICj(R);
38:      ICj(G)=Logisjic[1]^ICj(G);
39:      ICj(B)=Logisjic[2]^ICj(B);
40:      ICj(R)=ICj(B)^ICj-1(R);
41:      ICj(G)=ICj(R)^ICj-1(G);
42:      ICj(B)=ICj(G)^ICj-1(B);
43:}
```

4. Experimental Testing and Analysis

The experimental testing and analysis platform of this algorithm:

(1) Windows platform, CPU: Intel Core I7-8700、6 core、3.2GHz, RAM:8GB, with Windows10 OS, OpenCV 3.2, Visual Studio 2017. Image encryption security evaluation system was developed in C++.

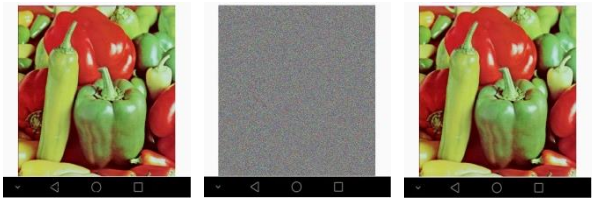
(2) Android Smartphone platform, CPU: Kirin 970、8 core、2.36GHz, RAM:6GB, with Android 9.0 OS. Test and analysis of image encryption and decryption effect and time are for the Android platform.

Using this algorithm, the effect of encrypting and decrypting Peppers (384×512) and Baboon (256×320) images on a mobile phone is shown in Figure 2. It can be seen from the figure that the encrypted image does not display the information of the original image, and decryption is performed after normal recovery of the original image.

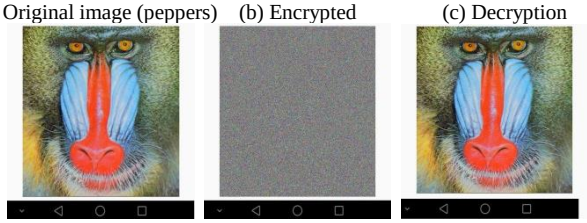
4.1. Histogram Analysis

The histogram of the image describes the distribution of pixel values in the image and is an important statistical feature of the image. Comparisons of the histograms of the *R*, *G* and *B* channels before and after encrypting the Peppers (384×512) image are shown in Figure 3. Figures 3(a), (c), (e) are the histograms of the red, green, and blue channels before the

original image is encrypted. It can be seen from the figure that the pixel distribution of the image has a large gap and is quite confusing. The attacker uses the pixel distribution of the image. It is easy to get the image information. Figures 3(b), (d), (f) are the histograms of the red, green, and blue channels obtained after the original image is encrypted. It can be seen from the figure that the distribution of the encrypted image pixels is relatively flat and uniform and can be well hidden. So, pixel value information can effectively resist statistical method attacks.

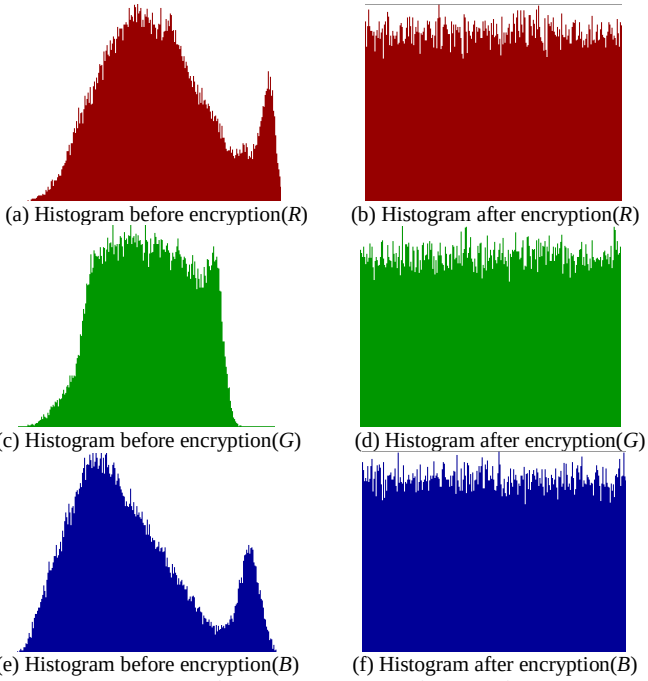


(a) Original image (peppers) (b) Encrypted (c) Decryption



(e) Original image (Baboon) (f) Encrypted (g) Decryption

Figure 2. Mobile phone encryption and decryption effect



(a) Histogram before encryption(*R*) (b) Histogram after encryption(*R*)

(c) Histogram before encryption(*G*) (d) Histogram after encryption(*G*)

(e) Histogram before encryption(*B*) (f) Histogram after encryption(*B*)

Figure 3. Comparison of channel histograms before and after encryption

4.2. Key Space Analysis

In this algorithm, the accuracy of the Cat map Key, Key[0], and Key[1] are 10^{-16}. The precision of ChebyShev[0-2] and Logistic[0-2] are 3×10^{16}. The total key space reaches $10^{16}\times10^{16}\times10^{16}\times(3\times10^{16})\times(3\times10^{16})=9\times10^{80}$, and the algorithm has a large enough key space, meaning it can effectively withstand exhaustive attacks.

4.3. Key Sensitivity Analysis

Using Lena(256×256) image as example, the right keys are Key[0] = 0.33, Key[1] = 0.33, and by increasing the two keys by 0.0000000000000001 (1×10^{-16}) to Key[0] = Key[1] = 0.3300000000000001, the decrypted image is very different from the original image. Figure 4(a) is an image of a ciphertext image decrypted with a correct key, and Figure 4(b) is an image of a decrypted key with a key that has had its value subtlety changed. From Figure 4(a) and Figure 4(b), it can be clearly seen

from the image contrast that when the initial key takes a small change of 10^{-16}, the original image cannot be recovered after the image is decrypted, which proves that the algorithm has good key sensitivity.

(a)Correct decrypted (b) After Changing Key

Figure 4. Key sensitivity analysis

4.4. Pixel Correlation Analysis

There is a great correlation between image pixels in the horizontal direction, vertical direction and diagonal direction. The attacker can use the adjacent pixel correlation of the ciphertext image to perform statistical attacks and analyze the original image. Therefore, the correlation of adjacent pixels is an important index for evaluating the security of the encryption algorithm. The average correlation coefficient comparison is shown in Table 1.

Table 1. Average correlation coefficient comparison (absolute value)

Figure			horizontal	vertical	diagonal
Cameraman(256×256)	Original image	Red	0.9307	0.9614	0.9166
		Green	0.9307	0.9614	0.9166
		Blue	0.9307	0.9614	0.9166
	Encrypted image	literature [3]	0.0242	0.0261	0.0245
		this algorithm	0.0213	0.0203	0.0217
Lena(256×256)	Original image	Red	0.9553	0.9186	0.9213
		Greed	0.9463	0.9409	0.9257
		Blue	0.9540	0.9514	0.8748
	Encrypted image	literature [14]Red	0.0996	0.0544	-
		this algorithm	0.0242	0.0153	0.0275
		literature [14]Green	0.0698	0.0529	-
		this algorithm	0.0016	0.0166	0.0223
		literature [14]Blue	0.0543	0.0829	-
		this algorithm	0.0059	0.0142	0.0051
Flower(787×576 787*576)	Original image	Red	0.9762	0.9672	0.9714
		Green	0.9867	0.9843	0.9802
		Blue	0.9876	0.9839	0.9793
	Encrypted image	literature [15]	0.0412	0.1251	0.0611
		this algorithm	0.0165	0.0284	0.0197

The formula for calculating the correlation coefficient between adjacent pixels in an image is shown in Equations (9)-(12):

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{9}$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N}[x_i - E(x_i)] \tag{10}$$

$$\mathrm{cov}(x, y) = \frac{1}{N}\sum_{i=1}^{N}[x_i - E(x_i)][y_i - E(y_i)] \tag{11}$$

$$r_{xy} = \frac{\mathrm{cov}(x, y)}{\sqrt{D(x)}\sqrt{D(y)}} \tag{12}$$

We randomly select 3000 pairs of neighboring pixels in the Cameraman, Lena, Flower, and ciphertext images, and calculate the average pixel correlations in the horizontal, vertical, and diagonal directions according to Equations (9)-(12), and then compare it to the literature [3, 15-16], as shown in Table 1. From Table 1, we can see that the

correlation of adjacent pixel pairs in the three directions are highly correlated. After the algorithm is encrypted in this paper, the correlation coefficient values of the adjacent pixel pairs in the three directions are very small, and the correlation is very low. The average correlation coefficient of the three channels of the Cameraman image encryption algorithm is similar to the algorithm in the literature [3], but the algorithm in this paper is still superior [3]. This algorithm and the literature [15-16] have compared the average correlation coefficient of Lena and Flower after encrypting the three channels, and the adjacent pixels of this algorithm have lower correlation. Therefore, the algorithm of this paper spreads the statistical characteristics of the original image into a random ciphertext very well, and the invisibility of the plaintext is enhanced, which shows that the algorithm can effectively resist statistical analysis.

4.5. Differential Attack Analysis

Differential analysis analyze the relationship between the ciphertext images before and after the change by a small change in the pixel values in the plaintext image. Differential attack analysis has two important indicators:

(1) The pixel change rate, NPCR, is used to test the number of ciphertext changes, as shown in Equation (13).

$$NPCR = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} D(i, j) \times 100\% \tag{13}$$

(2) The average pixel change density, UACI, is used to test the average density of ciphertext changes, as shown in Equation (14).

$$UACI = \frac{1}{M \times N} \left\{ \sum_{i,j} \frac{\left| C_1(i, j) - C_2(i, j) \right|}{255} \right\} \times 100\% \tag{14}$$

According to Equations (13) and (14), the NPCR and UACI of the three channels were calculated. The calculation results are shown in Table 2.

We use Lena, Flower, and Baboon images as tests. From the data in Table 2, we can see that for the Lena and Flower images, the NPCR and UACI values of the algorithm and the literature [2] in the red, green and blue channel are close, but this algorithm is slightly better than that of the literature [2]. For Flower images, the average NPCR in the literature [16] is 51.73% and UACI is 13.22%. The average NPCR of this paper is 99.62% and UACI is 33.52%. For the Baboon image, the average NPCR in the literature [7] is 73.68%, and the UACI is 19.85%. The average NPCR of the algorithm in this paper is 99.61% and the UACI is 33.48%. Therefore, the diffusion effect after a tiny change of the pixel value in the plaintext image is very good after the encryption of the algorithm, and the algorithm can effectively resist differential attack analysis.

Table 2. Three channel NPCR and UACI values

Image	Algorithm	NPCR(%)			UACI(%)		
		R	*G*	*B*	*R*	*G*	*B*
Lena	[2]	99.65	99.62	99.58	33.48	33.41	33.34
	Ours	99.65	99.63	99.60	33.50	33.42	33.51
Flower	[2]	99.62	99.60	99.55	33.39	33.52	33.40
	[15]	Average 51.73			Average 13.22		
	Ours	99.63	99.61	99.63	33.62	33.52	33.43
Baboon	[7]	Average 73.68			Average 19.85		
	Ours	99.60	99.61	99.61	33.45	33.53	33.47

4.6. Information Entropy Attack Analysis

Information entropy reflects the uncertainty of information. The higher the information entropy, the more chaotic the system. Information entropy calculation formula is shown as (15):

$$H(x) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \tag{15}$$

Calculate the entropy of the three channels of the image before and after encryption according to Formula (15). The calculation results are shown in Table 3.

Table 3. Comparison of information entropy before and after image encryption

Image		Red	Green	Blue
Lena	Before encryption	7.6412	7.6304	6.9892
	After encryption([6])	7.9891	7.9893	7.9893
	After encryption([14])	7.9966	7.9971	7.9967
	After encryption([15])	7.9948	7.9948	7.9948
	After encryption(Ours)	7.9992	7.9993	7.9992
Baboon	Before encryption	7.6323	7.3481	7.6672
	After encryption(Ours)	7.9977	7.9982	7.9979

For digital images, the closer the information entropy is to 8, the more random the image information becomes. From Table 3, we can see that the information entropy values of the red, green, and blue channel components after Lena and Baboon images that are encrypted by this algorithm are greater than the information entropy values before encryption, and the information entropy values of the three channels after Lena image algorithm encryption are higher than that of the literature [6, 15-16]. Furthermore, the entropy of the algorithm after encryption in this paper is closer to 8, indicating that the randomness of the pixel sequence of the encrypted image is very good.

4.7. Time Analysis

The most direct experience for users of image encryption and decryption algorithms is operational efficiency. In order to verify that the algorithm's execution efficiency is applicable to the Android platform, under the same operating environment, Lena(256 × 256), Girl(256 × 256), Peppers(384 × 512), Baboon(512 × 512), Barbara(787 × 576), Flower(800 × 800, 1200 × 800), Scenery(1200 × 1200, 1920 × 1200), and other images of different sizes were tested 100 times respectively, and then the average of the encryption/decryption time was recorded. The results are shown in Table 4. From Table 4, we can see that when the image pixels are 256 × 256, the algorithm of this paper is 0.005 seconds and 0.001 seconds longer than the encryption and decryption time of the algorithm of the literature [7], respectively. This is mainly because the algorithm has scrambled the image, while the literature [7] has not. When the image pixels are 512 × 512, 787 × 576, the encryption and decryption time of the algorithm is significantly less than the literature [7]. The encryption and decryption efficiency of this algorithm is obviously better than those proposed by [6, 15-16]. When the image size increases from 787 × 576 to 1200 × 1200, the 1200 × 1200 image size is 3.17 times that of the 787 × 576 image, the encryption time is 3.15 times the 787 × 576 image, and the decryption time is 3.11 times the 787 × 576 image. The time required for the algorithm's encryption and decryption increases significantly, mainly because the image pixel matrix becomes larger, the scrambling overhead increases, and the XOR operation of the pixel point increases. But, the encryption time is within 0.8 seconds, which is obviously better than other algorithms. When the image pixels increase from 1200×1200 to 1920 × 1200, the image size increases by 60%, the encryption time increases from 0.799 seconds to 1.087 seconds, an increase of 36%, and the decryption time increases from 0.783 seconds to 1.026 seconds, an increase of 31%. With increasing image pixels, the increase in time is slower because the number of scrambling times does not increase significantly despite the increase in the number of pixels, and the number of Logistic and Chebyshev iterations do not increase. For large images, the decryption time is about 1 second, and its speed is also significantly better than other algorithms of the same type. Experiments show that the algorithm has faster encryption and decryption speed and can be widely applied to image encryption of smart mobile devices such as mobile phones and tablet computers with high real-time requirements.

Table 4. Encryption and decryption time test results

Image	Algorithm	Encryption time (s)	Decryption Time (s)
Lena(256 × 256)	[6]	0.267	-
	[14]	0.261	0.269
	Ours	0.051	0.049
Girl(256 × 256)	[7]	0.047	0.050
	Ours	0.052	0.051
Peppers(384 × 512)	[15]	3.68	2.97
	Ours	0.118	0.104
Baboon(512 × 512)	[6]	1.370	-
	[7]	0.181	0.180
	Ours	0.148	0.143
Barbara(787 × 576)	[7]	0.306	0.312
	Ours	0.254	0.251
Flower(800 × 800)	Ours	0.353	0.346
Flower(1200 × 800)	Ours	0.531	0.521
Scenery(1200 × 1200)	Ours	0.799	0.783
Scenery(1920 × 1200)	Ours	1.087	1.026

5. Conclusions

In this paper, an image encryption algorithm based on three kinds of chaotic maps is designed. The algorithm uses the Cat mapping method to calculate the optimal iteration number to fully scramble the image. Generating the initial key by taking the random number and the mutual mapping of Logistic and ChebyShev improves the key space greatly. Using Logistic and ChebyShev double chaotic mapping methods, the pixel differences between adjacent pixels are larger, and the correlation between pixel points is destroyed, making the correlation of the adjacent pixels of the image after encryption reduced. The pixel R', G', and B' components of the pixel at the previous position of the encrypted pixel are used to perform exclusive-OR operation with the *B*, *R*, and *G* components of the pixel being encrypted, respectively, so that when the plaintext changes a little bit, the plaintext images are completely different from the encrypted ones. So, it can greatly improve the image's ability to resist different attacks. Reducing the Cat, Logistic, and ChebyShev iteration number of iterations on the premise of ensuring security can reduce encryption time, improve the efficiency of encryption, and allow this algorithm to be widely used in image encryption of smart mobile devices based on the Android platform that requires high real-time performance.

Acknowledgments

This work is partly supported by the Science and Technology Plan Project of Hunan Province (2016TP1020), the Open Fund Project of Hunan Provincial Key Laboratory of Intelligent Information Processing and Application for Hengyang Normal University (IIPA18K03), the Hunan Provincial Natural Science Foundation of China (Grant No. 2017JJ2010, 2017JJ4001), the Scientific Research Fund of Hunan Provincial Education Department (Grant No. 16B039), the Application-oriented Special Disciplines, and the Double First-Class University Project of Hunan Province(Xiangjiaotong [2018] 469).

References

1. C. Roberta, G. Werner, and E. Fernando, "SWOT: Google Device Technologies, Worldwide," (https://www.gartner.com/doc/3896269?ref=mrktg-srch, accessed December 2018)
2. C. Pak and L. Huang, "A New Color Image Encryption using Combination of the 1D Chaotic Map," *Signal Processing*, Vol. 138, pp. 129-137, 2017
3. M. J. Rostami, A. Shahba, S. Saryazdi, and H. Nezamabadi-pour, "A Novel Parallel Image Encryption with Chaotic Windows based on Logistic Map," *Computers and Electrical Engineering*, Vol. 62, pp. 384-400, 2017
4. V. Patidar, N. K. Pareek, G. Purohit, and K. K. Sud, "Modified Substitution Diffusion Image Cipher using Chaotic Standard and Logistic Maps," *Communications in Nonlinear Science and Numerical Simulation*, Vol. 15, No. 10, pp. 2755-3-2765, 2010
5. S. Tayde and S. Siledar, "File Encryption, Decryption using AES Algorithm in Android Phone," *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 5, No. 5, pp. 550-554, 2015
6. W. T. Zheng, "Image Encryption Research based on Android Phones," Central China Normal University, 2016
7. Y. H. Zhao, "Image Encryption APP Design and Implementation on the Android Platform," Chongqing Normal University, 2016
8. Z. Jiang, "Color Image Encryption based on Chaotic Mapping Cat Android," *Industrial Control Computer*, Vol. 30, No. 4, pp. 98-100, 2017
9. Q. C. Zhong, Q. X. Zhu, and P. L. Zhang, "Multiple Chaotic Maps Encryption System," *Journal of University of Electronic Science and Technology of China*, Vol. 38, No. 2, pp. 274-277, 2009
10. C. X. Zhu, D. Z. Huang, and Y. Guo, "A New Image Encryption Algorithm based on Multiple Chaotic Maps and Output-Feedback," *Geomatics and Information Science of Wuhan University*, Vol. 35, No. 5, pp. 528-531, 2010
11. G. Jiao, X. Peng, and K. Duan, "Image Encryption with The Cross Diffusion of Two Chaotic Maps," *KSII Transactions on Internet and Information Systems*, Vol. 13, No. 2, pp. 1064-1079, February 2019
12. H. Lai, M. A. Orgun, J. Pieprzyk, J. Li, M. X. Luo, J. H. Xiao, et al., "High-Capacity Quantum Key Distribution using Chebyshev Map Values Corresponding to Lucas Numbers Coding," *Quantum Information Processing*, Vol. 15, No. 11, pp. 4663-4679, 2016
13. J. M. Liu, S. S. Qiu, F. Xiang, and H. J. Xiao, "Information Encryption Algorithm based on Multiple Chaotic Mappings," *Journal of South China University of Technology* (*Natural Science Edition*), No. 5, pp. 1-5, 2007
14. Z. G. Huang, "Watermarking Algorithm based on Best Permutation," *Communications Technology*, Vol. 43, No. 3, pp. 150-152, 2010
15. W. Wang and C. Jin, "Image Encryption Scheme for Android Mobile Platform," *Computer Science*, Vol. 41, No. 8, pp. 94-96, 2014
16. B. B. Du, "Research and Implementation of an Image Encryption System based on Android," Wuhan University of Technology, 2013