Cryptanalysis of an ergodic chaotic encryption algorithm*

Wang Xing-Yuan(王兴元)†, Xie Yi-Xin(谢旖欣), and Qin Xue(秦 学)

Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116024, *China*

(Received 16 September 2011; revised manuscript received 10 October 2011)

In this paper, we present the results for the security and the possible attacks on a new symmetric key encryption algorithm based on the ergodicity property of a logistic map. After analysis, we use mathematical induction to prove that the algorithm can be attacked by a chosen plaintext attack successfully and give an example to show how to attack it. According to the cryptanalysis of the original algorithm, we improve the original algorithm, and make a brief cryptanalysis. Compared with the original algorithm, the improved algorithm is able to resist a chosen plaintext attack and retain a considerable number of advantages of the original algorithm such as encryption speed, sensitive dependence on the key, strong anti-attack capability, and so on.

Keywords: chaos, cryptanalysis, encryption, block-cipher

PACS: 05.45.Ac, 05.45.Gg, 05.45.Vx **DOI:** 10.1088/1674-1056/21/4/040504

1. Introduction

A chaotic system has many excellent features that are widely applied to various fields,[1−5] and these include its application in cryptography.[6−8] Baptista[9] proposed a new encryption system based on the ergodicity property of a chaotic system in 1998. As Baptista used the number of iterations as a cipher text, his encryption algorithm is helpful for avoiding a statistical attack and a differential attack, and this is also conductive to modern digital network transmission, but it also restricts the encryption speed. Later, analyses[10,11] on this cryptographic algorithm and improvement[12−19] of the program were published one after another. Álvarez *et al.*[20] made full-scale cryptanalysis about Baptista's algorithm and attacked it successfully by using many methods. On these bases, Wang[21] proposed a new fast chaotic encryption algorithm. After analysing the performance of this new algorithm, she claimed that this new algorithm can resist chosen plaintext attack. But in fact the chosen-plaintext attack can successfully break the algorithm. In this paper we first briefly introduce Wang's algorithm,[21] and then make a cryptanalysis and improvement of this algorithm.

2. Description of the original algorithm

(i) Divide the plaintext: set a byte as a block, such that each block corresponds to an ASCII code value ($0 \sim 255$).

(ii) Divide the chaotic attractor: from the attractor, select a fixed length of subinterval, which is named interval I in this paper. The random numbers come from this interval I only. We divide the interval into 256 sub-intervals with the same length. Each subinterval corresponds to an integer from 0 to 255. Then there is a map between the ASCII values and the outputs of the chaotic random numbers. In this paper, as an example, we take the logistic map as a chaotic map.

(iii) Encryption process: the parameter b and the initial point x_0 of the chaotic map are treated as secret keys. The smaller integer k is treated as the initial increment and the bigger integer s is treated as the initial iterative time, which are generated by b and x_0 according to some rules. For instance, s and k can be obtained as $s = \mathrm{int}(b/x_0 \times 50)$, $k = \mathrm{int}(b/x_0)$, where $\mathrm{int}(\cdot)$ means getting the integer number.

Starting the chaotic iteration: first, iterate the chaotic map s times initially. After variable x is iter-

*Project supported by the National Natural Science Foundation of China (Grant Nos. 61173183, 60573172, and 60973152), the Doctoral Program Foundation of Institution of Higher Education of China (Grant No. 20070141014), and the Natural Science Foundation of Liaoning Province, China (Grant No. 20082165).

†Corresponding author. E-mail: wangxy@dlut.edu.cn

© 2012 Chinese Physical Society and IOP Publishing Ltd

ated s times, if x does not fall in the attractor interval I, then we continue to iterate some k-times until x falls in the interval. The x must belong to a sub-interval of which the corresponding integer is w_1. Then w_1 plus the first letter's ASCII p_1 is $m_1 = w_1 + p_1$. If $m_1 < 256$, m_1 is treated as the cipher text c_1 of the first letter and put out; otherwise $c_1 = m_1 - 256$.

Next, use $s = 256 - c_{i-1}$, $(i = 2, 3, \ldots)$ as the initial iterative time of the i-th plaintext letter, and then compute x; if the x does not fall in the attractor interval I, then we continue to iterate some k-times until x falls in the interval, where $k = 256 - p_{i-1}$. The sub-interval that x falls in corresponds to an integer w_i. w_i plus the i-th letter's ASCII p_i is $m_i = w_i + p_i$. If $m_i < 256$, m_i is treated as the cipher text c_i of the i-th letter and put out; otherwise $c_i = m_i - 256$.

The flow chart of the original encryption algorithm is shown in Fig. 1 and an example of dividing the chaotic attractor is shown in Fig. 2, where s_i and k_i are expressed as

$$s_i = f(b, x_{i-1}, c_{i-1}) = \begin{cases} \lfloor b/x_{i-1} \times 50 \rfloor, & i = 1, \\ 256 - c_{i-1}, & i \geq 2, \end{cases} \quad (1)$$

$$k_i = g(b, x_{i-1}, p_{i-1}) = \begin{cases} \lfloor b/x_{i-1} \rfloor, & i = 1, \\ 256 - p_{i-1}, & i \geq 2. \end{cases} \quad (2)$$

(iv) Decryption process: this process is similar to the encryption process. Through comparing cipher c_i with w_i, if $c_i \geq w_i$, the plaintext ASCII value is $p_i = c_i - w_i$; otherwise, $p_i = c_i + 256 - w_i$. Therefore, the plaintext is recovered, and at the same time, the next s and k are computed.

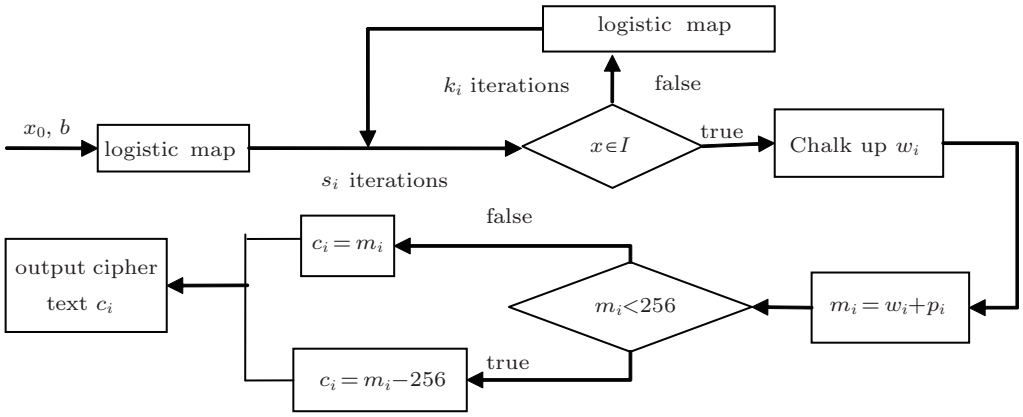


Fig. 1. Flow chart of the original encryption algorithm.

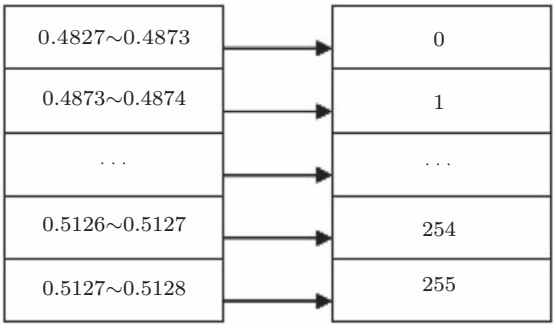


Fig. 2. Each sub-interval corresponds to an integer.

3. Chosen plaintext attack

3.1. Cryptanalysis

Pay attention to the fact that the values of w_1 in the encryption and the decryption are the same as those in the above algorithm, and the w_1 is related only to x_0 and b but has nothing to do with the plaintext, so w_1 can be obtained by chosen-plaintext attack. Furthermore, p_1 can be decrypted with the support of w_1, so that the first plaintext is disclosed.

When $i > 1$, w_i is related to the location of x which is obtained from the previous iteration, and also to the present iteration numbers of s_i and k_i. It seems that we cannot continue to decrypt because of lack of the location of x which is obtained from the previous iteration, but actually this problem can be solved in the chosen plaintext attack. There is no harm in using mathematical induction to prove the effectiveness of the chosen plaintext attack.

i) As mentioned above, we can obtain the first one plaintext p_1 from cipher text c_1.

ii) Assume that we can break the former $(i-1)$-th

cipher text c_{i-1} and then obtain the former $(i-1)$-th plaintext p_{i-1}.

iii) Choose the plaintext $p'_1, p'_2, \ldots, p'_{i-1}, p'_i$, where $p'_j = p_j, (j \leq i-1)$. We can temporarily use encryption equipment to obtain the corresponding cipher text $c'_1, c'_2, \ldots, c'_{i-1}, c'_i$ of the chosen plaintext in chosen plaintext attack. Suppose that x has been iterated to x'' when the i-th chosen plaintext is encrypted, and suppose x has been iterated to x' when the i-th original plaintext is encrypted. The former $i-1$ chosen plaintext p' and original plaintext p are the same, so the x' and x'' are the same.

When $i > 1$, the iterations s'_i and k'_i which can influence the value of the w'_i are related only to c'_{i-1} and p'_{i-1}. Furthermore, p'_{i-1} and p_{i-1} are chosen to be equal (then c'_{i-1} and c_{i-1} are equal). Thus the w'_i which is generated from chosen plaintext attack and the w_i which is generated from normal encryption are equal.

By comparing c'_i and p'_i we can obtain w'_i, and then have w_i, so that we can obtain plaintext p_i.

In summary, the chosen plaintext attack is effective.

3.2. Example

Set up plaintext for the "abc". When $x_0 = 0.52$, $b = 3.78$, and $I = [0.4872, 0.5128]$, the corresponding cipher text is "92 132 1".

Now we do not use keys but a chosen plaintext attack to restore the plaintext.

(I) Choose plaintext "d" with ASCII 100 (actually this choice could be arbitrary, and only requires that it has been known). Encrypt it with the encryption equipment as cipher text "95". That a positive integer w'_1 fails to be found makes $100 + w'_1 = 95$, so it can be concluded that $100 + w'_1 - 256 = 95$, then $w'_1 = 251$, namely $w_1 = 251$. Thus $p_1 = 92 + 256 - 251 = 97$, namely, the first plaintext we restored is the "a".

(II) Choose plaintext "ad" with ASCII 97 and 100. Encrypt it with the encryption equipment as cipher text "92, 134". We can find $w'_2 = 34$ to make $100 + w'_2 = 134$, so it can be concluded that $w'_2 = 34$, namely $w_2 = 34$. Thus $p_2 = 132 - 34 = 98$, namely the second plaintext we want to restore is the "b".

(III) Choose plaintext "abd" with ASCII 97, 98, and 100. Encrypt it with the encryption equipment as cipher text "92, 132, 2". That a positive integer w' fails to be found makes $100 + w'_3 = 2$, so it can be concluded that $100 + w'_3 - 256 = 2$, then $w'_3 = 158$, namely $w_3 = 158$. Thus $p_3 = 1 + 256 - 158 = 99$, namely, the third plaintext that we want to restore is the "c".

4. Improvement

Although this algorithm is vulnerable to the chosen plaintext attack, it extends the advantages of Baptista's algorithm at the same time, and its speed is much faster. These advantages in the literature[21] have been discussed, so we will not mention them here, but instead make additional improvements.

4.1. Analysis

There are three main reasons why the chosen plaintext attack can be efficient.

I) There is a fatal security problem in the algorithm design, that is, w can be deduced by the known c and p. Since the operations among c, p, and w are simple add operations, w can be deduced by the value comparisons and plus-minus operations if cipher text c and plaintext p have been known.

II) The w_1 is concerned only with the key and not the plaintext, which makes it possible that w_1 can be deduced by an arbitrary chosen plaintext. This is because that arbitrary chosen plaintext and original plaintext can obtain the same w_1.

III) Since w_i $(i > 1)$ is affected only by the previous $i-1$ cipher text but not by p_i and the subsequent plaintext, so the follow-up w_i can be deduced by w_1. The chain of causation can be expressed as $w_1 \xrightarrow{c_1} p_1 \xrightarrow{c_2} p_2 \to w_3 \to \cdots \to w_i$.

Therefore, there are also three corresponding improvement directions:

(A) improve the operations among c, p, and w, which makes it difficult or impossible to deduce w by known c and p;

(B) set w_1 to be affected by plaintext;

(C) set w_i $(i > 1)$ to be affected by p_i or the subsequent plaintext.

4.2. Measures

A) Modify the operations among c, p, and w: Suppose that it costs $s_i + k_i \times v$ iterations from previous w to current w. First take p rotate left v-bit to obtain temporary variable t, then do XOR operation between t and w to have cipher text c.

From the features of the chaotic map we can see that v and w are random. So even using the chosen

plaintext attack and knowing c and p, one cannot accurately derive v and w. If one wants to enumerate the chosen-plaintext attack, only the derivation of w would cost at least 8 enumerations (there are 8 possible situations when p is rotated left). It will cost at least 8^n enumerations to decrypt a plaintext whose length is n, not including the cost of choosing plaintext and encrypting plaintext so many times.

B) Use multiple rounds of encryption. Improved by the above step the algorithm may still be attacked by the combination of the chosen plaintext attack and enumeration attack. But after doing more than two rounds of encryptions, we cannot find any direct arithmetical relationship between c and p, so that the chosen plaintext attack becomes invalid. Here, we take two rounds for example.

C) The above two improvements are for the fact that the improving of security is unavoidably at the expense of speed. But we can strive to reduce this expense. In order to reduce the expense of speed, we will replace the expressions of s and k as

$$s_i = f(b, x_{i-1}, c_{i-1}) = \begin{cases} \lfloor b/x_{i-1} \times 50 \rfloor, & i = 1, \\ c_{i-1}\%256, & i \geq 2, \end{cases} \quad (3)$$

$$k_i = g(b, x_{i-1}, p_{i-1}) = \begin{cases} \lfloor b/x_{i-1} \rfloor, & i = 1, \\ p_{i-1}\%256 + 1, & i \geq 2. \end{cases} \quad (4)$$

Experiments show that with using only the first two improvements, it takes 12.78 s to encrypt a 44.1 k text file, but with additionally combining the third improvement, it will take only 6.9 s to encrypt the same text file.

4.3. Flow chart after improvement

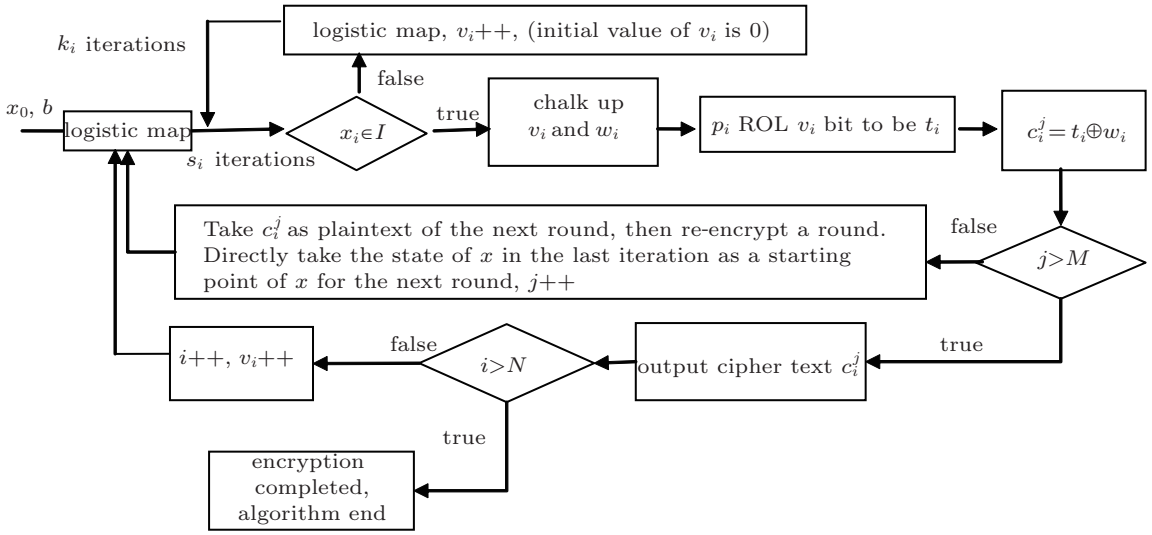


Fig. 3. Flow chart of the improved algorithm. Annotation: the M in this graph is a pre-set number of encryption rounds. p is the plaintext. c is the cipher text. Subscript i indicates the i-th plaintext that is being encrypted. Superscript j indicates the present j-th round of encryption. Except the first iteration to take the key x_0 as a starting point, each subsequent iteration takes the end of the last iteration as the beginning of the present iteration.

4.4. Example of the improved algorithm

In order to demonstrate our improved algorithm, we suppose that the plaintext is "abc", $x_0 = 0.43203125$, $b = 3.78$, the attractor interval $I = [0.4872, 0.5218]$, then the cryptograph of "abc" is "111 114 1".

The process is as follow: $s_1 = \text{int}(b/x_0 \times 50) = 437$ and $k_1 = \text{int}(b/x_0) = 8$. After chaotically iterating 437 times, x is 0.2917 that is not in the interval $I = [0.4872, 0.5218]$. So we continue to iterate $217 \times k_1$ times and obtain $x = 0.51158$ that is in the selected attractor interval. The corresponding integer w_1^1 of the sub-interval that x belongs to is 243. The ASCII p_1 of "a" is 97. Then we can obtain $v_1^1 = 217$, $t_1^1 = \text{leftshift}(p_1, v_1^1) = 194$, and $c_1^1 = t_1^1 \oplus w_1^1 = 49$, where $\text{leftshift}(p_1, v_1^1)$ means letting p rotate left v_1^1-bit. The next is the second loop of encryption. After chaotically iterating 437 times, x is 0.291479 that is not in the interval. So we continue to iterate $76 \times k_1$ times and obtain $x = 0.499619$ that is in the selected attractor interval. The corresponding integer w_1^2 of

the sub-interval that x belongs to is 124. Then we can obtain $v_1^2 = 76$, $t_1^2 = \text{leftshift}(c_1^1, v_1^2) = 19$, and $c_1 = t_1^2 \oplus w_1^2 = 111$.

Then we compute s once again, and obtain $s_2 = c_1\%256 = 111$, $k_2 = p_1\%256 + 1 = 98$. After chaotically iterating 111 times, x is 0.920194, which is not in the attractor interval. So we continue to iterate $26 \times k_2$ times and obtain $x = 0.503268$, which is in the selected attractor interval. The corresponding integer w_2^1 of the sub-interval that x belongs to is 160. The ASCII p_2 of "b" is 98. Then we can obtain $v_2^1 = 26$, $t_2^1 = \text{leftshift}(p_2, v_2^1) = 137$, and $c_2^1 = t_2^1 \oplus w_2^1 = 41$. The next is the second loop of encryption. After chaotically iterating 111 times, x is 0.221058, which is not in the interval. So we continue to iterate $2 \times k_2$ times and obtain $x = 0.508651$, which is in the selected attractor interval. The corresponding integer w_2^2 of the sub-interval that x belongs to is 214. Then we have $v_2^2 = 2$, $t_2^2 = \text{leftshift}(c_2^1, v_2^2) = 164$, and $c_2 = t_2^2 \oplus w_2^2 = 114$.

According to the same principle, the cipher text c_3 of "c" is obtained to be 1. Therefore the cryptograph of "abc" is "111 114 1".

5. Cryptanalysis after improvement

1) Encryption speed: as a result of more complex computing and two rounds of encryption, the improved algorithm is a little slower than the original one, but it still has many advantages over Baptista's algorithm. The author encrypted 44.1 k text file with $b = 3.78$, $x = 0.43203125$ and the selected attractor interval $I = [0.4872, 0.5218]$ in the following experimental environment: CPU: AMD Athlon(tm) 64 Processor 3000+; Memory: 768M; Operating System: Windows XP; Programming Language: VC++. The speeds of the original algorithm and the improved one are shown in Table 1.

In fact there is close relationship between the speed of encryption and the selection of attraction subinterval, Table 2 shows the speeds of encryption when the selected subinterval is $I = [0.4, 0.6]$:

Table 1. Encryption times of the three algorithms when $I = [0.4872, 0.5218]$.

	Original algorithm	Implement the improvement of i and ii	Implement the improvement of i, ii, and iii
Encryption time/s	6.704	12.75	6.9

Table 2. Encryption time of the three algorithms when $I = [0.4, 0.6]$.

	Original algorithm	Implement the improvement of i and ii	Implement the improvement of i, ii, and iii
Encryption time/s	0.734	1.375	0.875

2) Key space: the key of the improved algorithm is the initial value x_0 and control parameter b. They can be any values which let the system be a chaotic system within the accuracy of the computer.

3) Sensitivity: when the secret keys undergo a minor change, the cipher text changes dramatically. Figure 4(a) shows the cipher text of encrypting alphabetical 26 lowercase letters with b changing 0.00001, where the horizontal axis coordinates mark 26 plaintext characters and the vertical coordinates mark cipher text. Figure 4(b) shows the cipher text of encrypting alphabetical 26 lowercase letters with x changing 0.00001. In Fig. 4, the dashed and the solid lines, respectively, denote the cipher text before and after the secret key is changed. It can be seen from Fig. 4 that the cipher text changes evidently when the key changes slightly. It means that the improved algorithm retains the high sensitivity of the original algorithm to the parameters change.

4) Chosen plaintext attack: as discussed in Subsection 4.2, the v and w are random and have the features of a chaotic map. So even using the chosen plaintext attack and knowing c and p, one cannot accurately derive v and w. If one wants to enumerate them by using a chosen-plaintext attack, only the derivation of w would cost at least 8 enumerations (there are 8 possible situations when p is rotated left). It will cost at least 8^n enumerations to decrypt a plaintext whose length is n, which does not include the cost of choosing plaintext and encrypting plaintext so many times.

5) Statistical characteristic: the ASCII value distribution of the first 3000 plaintext letters in a section of an English article and the corresponding cipher texts are shown in Figs. 5(a) and 5(b) respectively. From Fig. 5, we can know that the ASCII value before encryption is mainly between more than 90 and more than 120, but cipher text ASCII value is distributed uniformly after encryption, which makes the attack difficulty high greatly.

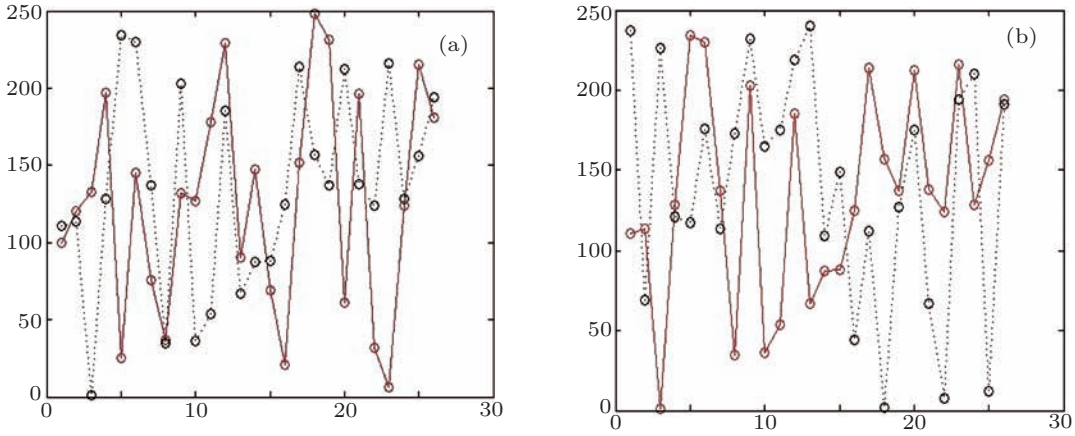


Fig. 4. (colour online) Results of sensitivity analysis of the secret key with b (a) and x (b) changing 0.00001.

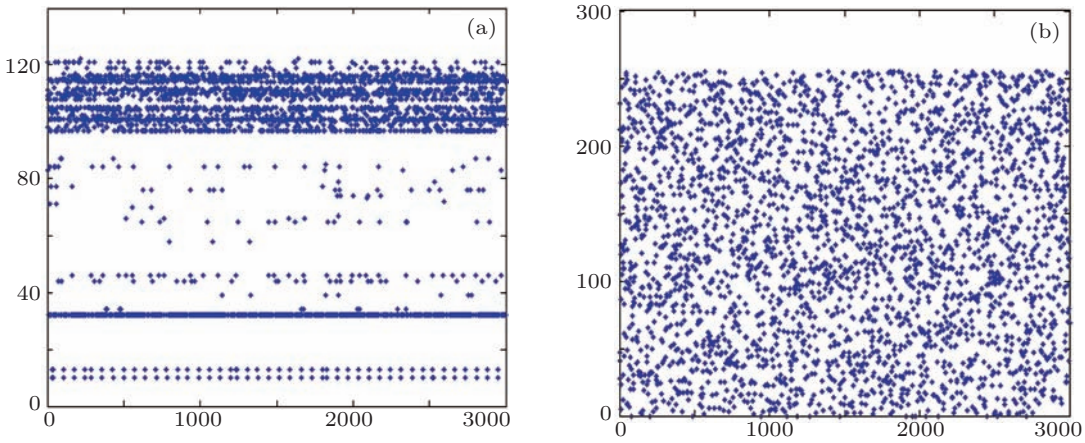


Fig. 5. (colour online) Distributions of plaintext ASCII values (a) and the cipher ASCII values (b) distribution of the same plaintext.

6. Conclusion

In this paper, we analysed a new faster chaotic encryption algorithm, which was proposed by Wang Xiao-Juan *et al.*,[21] and successfully break their algorithm by using the chosen plaintext attack. On the basis of the cryptanalysis we improve their algorithm and perform a brief cryptanalysis of the improved algorithm. Compared with the original algorithm, the improved algorithm is able to resist the chosen plaintext attack, and retain a considerable number of advantages of the original algorithm, such as encryption rapidness, high sensitivity to the secret key, resistance to many attacks, and so on. Currently, chaos cryptography is developing rapidly. The improved algorithm proposed in this paper can be said to be faster and safer only in the present situation. With the further development of chaotic cryptography, the improved algorithm proposed in this paper also needs to be further studied and improved.

References

[1] Wang X Y 2003 *The Chaos in Complex Non-linear Systems* (Beijing: Electronic Industry Press) p. 5 (in Chinese)

[2] Zhang H G, Xie Y H, Wang Z L and Zheng C D 2007 *IEEE Trans. Neural Networks* **18** 1841

[3] Ma D Z, Zhang H G, Wang Z S and Feng J 2010 *Chin. Phys.* B **19** 050506

[4] Zhang H G, Ma D Z, Wang Z S and Feng J 2010 *Acta Phys. Sin.* **59** 147156 (in Chinese)

[5] Zhang H G, Ma T D, Huang G B and Wang Z L 2010 *IEEE Trans. Sys., Man and Cybernetics* **40** 831

[6] Wang X Y, Qin X and Xie Y X 2011 *Chin. Phys. Lett.* **28** 080501

[7] Wang J and Jiang G P 2011 *Acta Phys. Sin.* **60** 060503 (in Chinese)

[8] Jin J X and Qiu S S 2010 *Acta Phys. Sin.* **59** 792 (in Chinese)
[9] Baptista M S 1998 *Phys. Lett.* A **240** 50
[10] Jakimoski G and Kocarev L 2001 *Phys. Lett.* A **291** 381
[11] Alvarez G, Montoya F, Romera M and Pastor G 2004 *Phys. Lett.* A **326** 211
[12] Wong W K, Lee L P and Wong K W 2001 *Comput. Phys. Commun.* **138** 234
[13] Wong K W 2002 *Phys. Lett.* A **298** 238
[14] Palacios A and Juarez H 2002 *Phys. Lett.* A **303** 345
[15] Wong K W 2003 *Phys. Lett.* A **307** 292
[16] Li S, Chen G, Wong K W, Mou X and Cai Y 2004 *Phys. Lett.* A **332** 368
[17] Wang X Y, Duan C F and Gu N N 2008 *Int. J. Mod. Phys.* B **22** 901
[18] Li H J and Zhang J S 2010 *Chin. Phys.* B **19** 050508
[19] Wang X Y and Xie Y X 2011 *Chin. Phys.* B **20** 080504
[20] Álvarez G, Montoya F, Romera M and Pastor G 2003 *Phys. Lett.* A **311** 172
[21] Wang X Y and Wang X J 2011 *Int. J. Mod. Phys.* B **25** 2047