Cryptanalysis of a chaos-based cryptosystem with an embedded adaptive arithmetic coder*

Wang Xing-Yuan(王兴元)† and Xie Yi-Xin(谢旖欣)

School of Electronic & Information Engineering, Dalian University of Technology, Dalian 116024, *China*

(Received 16 September 2010; revised manuscript received 2 April 2011)

In this paper, we analyse a new chaos-based cryptosystem with an embedded adaptive arithmetic coder, which was proposed by Li Heng-Jian and Zhang J S (Li H J and Zhang J S 2010 *Chin. Phys.* B **19** 050508). Although this new method has a better compression performance than its original version, it is found that there are some problems with its security and decryption processes. In this paper, it is shown how to obtain a great deal of plain text from the cipher text without prior knowledge of the secret key. After discussing the security and decryption problems of the Li Heng-Jian *et al.* algorithm, we propose an improved chaos-based cryptosystem with an embedded adaptive arithmetic coder that is more secure.

Keywords: chaos, cryptography, compression, arithmetic coding

PACS: 05.45.–a, 05.45.Gg, 05.45.Vx **DOI:** 10.1088/1674-1056/20/8/080504

1. Introduction

Since Baptista proposed an ergodic chaotic encryption algorithm in 1998,[1] chaotic cryptography research has aroused more and more attention.[2−5] In view of the inadequacy of the Baptista algorithm, a number of researchers made cryptanalysis[6,7] against it and then proposed some improvements.[8−12] Especially in recent years, a constant rise in the amount of information, as well as the need for security, has led to a growing interest in the investigation of the joint operation of compression and encryption on the same set of data. Therefore, a number of chaotic cryptosystems with an embedded coder have been proposed recently.[13−15] However, most of these algorithms focus on entropy coding and transmission coding compression structures, in which the chaotic system only plays the role of a pseudo-random number generator, instead being based on real system chaos architecture.

Recently, a new chaos-based cryptosystem with an embedded adaptive arithmetic coder was proposed by Li Heng-Jian *et al.*[16] It was an improved version of similar cryptography proposed by Kwok-Wo Wong *et al.*[17] and obtained a better compression performance by introducing an adaptive mechanism and using arithmetic coding[18−20] instead of Huffman coding, etc. However, there are some problems with both its security and its decryption process. Therefore, this paper analyzes the shortcomings of this algorithm and proposes an improved chaos-based cryptosystem with an embedded adaptive arithmetic coder that has better security.

2. Brief description of the Li Heng-Jian *et al.* algorithm

Wong *et al.* proposed an improved Baptista algorithm in their chaos-based cryptosystem. Figure 1 shows an example of the look-up table used in the improved Baptista algorithm. Based on Wong *et al.*'s algorithm, Li Heng-Jian *et al.* used the dynamic adaptive look-up table instead of the static one devised by Wong *et al.* The dynamic adaptive look-up table is initialized by setting the occurrence number of each source of symbols to 1 and then is updated at regular intervals according to the statistical information of the symbols which have appeared already. This kind of adaptive look-up table neither needs to scan the whole plain text in advance, nor needs additional transmission of statistical information, so it is more realistic and also more convenient to use.

*Project supported by the National Natural Science Foundation of China (Grant Nos. 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 of China (Grant No. 20082165).

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

© 2011 Chinese Physical Society and IOP Publishing Ltd

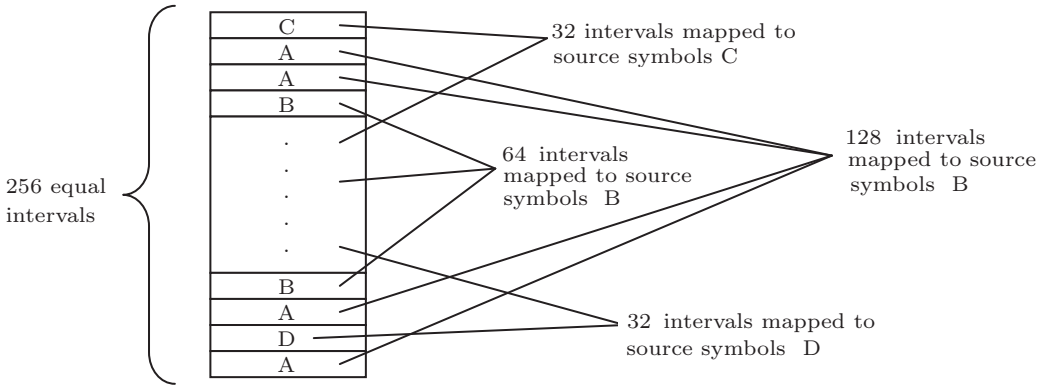


Fig. 1. The sketch map of the look-up table of Wong *et al.*'s algorithm.

2.1. Encryption

Figure 2 shows the encryption flowchart of Li Heng-Jian *et al.*'s algorithm. In the figure, K_1 and K_2 are two chaos sequences generated by two different chaos maps, which are chosen as the logistic map and the Tent map, respectively. However, in the actual programme, K_1 and K_2 are preserved as control parameters. The specific encryption procedure is as follows:

(i) For no initial information at the beginning, initially set all of the occurrence numbers of the symbols to the same number, such as 1. Then select the most M probable symbols, for example the first 128 ASCII in 256 ASCII where $M = 128$.

(ii) Build the look-up table according to the probable M characters and their probability dynamically. Firstly, divide the phase space into N equal-width intervals and then map the probable M characters into N equal-width intervals by the following formula where j ranges from 1 to N:

$$n(s_j) = \left\lfloor \frac{N \times u(s_j)}{\sum_{i=1}^{M} u(s_i)} \right\rfloor + 1, \tag{1}$$

in which $u(s_j)$ is the occurrence number of the source symbol s_j, $n(s_j)$ is the amount of phase space intervals mapped to source symbol s_j. The plain text character is encrypted as the number of iterations applied in the look-up tables.

(iii) Dividing the plain text into two groups according to the occurrence frequency of plain text characters. One group, which is named Group 1, includes the plain text characters corresponding to the selected M high probability source symbols. The other one, which is named Group 2, includes the remainder. The plain text in Group 1 is encrypted by the improved Baptista algorithm mentioned in the introduction, while the plain text in Group 2 is encrypted by the way of XOR. If the number of iteration has reached the maximum iteration, which is predetermined as MAX, put this plain text character into Group 2 while still insert "MAX" as a mark into the cipher text of Group 1. In order to facilitate decryption, the position of the cipher text sequence obtained from Group 1 should correspond to the position of the plain text sequence also obtained from Group 1. That is to say, where the plain text character is in the plain text, so is the cipher text character in the cipher text. In order to preserve the relative position, insert "$MAX + 1$" as a mark to the corresponding position in the cipher text for the plain text character which is not in the Group 1.

(iv) Compress the cipher text of Group 1 by using adaptive arithmetic coding and then name these cipher texts, which have been compressed as a compressed sequence. Name the cipher texts of Group 2 as an XOR sequence since they are processed by an XOR operation, and name the cipher texts of Group 1, which have not been yet compressed as a Baptista sequence since they are processed by the Baptista algorithm. Then mix the compressed sequence and the XOR sequence to become a mixed sequence S by turns, that is to say if one in the mixed sequence is from a compressed sequence then the next one should be from the XOR sequence and *vice versa*. If one of the two sequences is longer, add the superabundance of the longer one to the end of the mixed sequence S straightway.

(v) Count the plain texts that have been processed in order to update the probability of the source symbols if the predetermined cycle period is achieved. Select afresh M high probability source symbols and then go to Step (ii) to update the look-up table.

(vi) Process the mixed sequence S by the following diffusion operation when all of the plain texts have been processed.

$$C_i = f(P_i, K_i, \gamma) = (P_i \oplus K_i + AC_{i-1}) \bmod 256, \quad (2)$$

where γ is an initial cipher text $C_0 = \gamma$ related to the key, P_i is the mixed sequence mentioned above, A is a diffusion factor and K_i is a key sequence achieved by the following formula:

$$K_i = \text{round}(F_\alpha(x_{i-1}) \times 2^{31}) \bmod 256, \quad i = 1, 2, \ldots,$$

where $F_\alpha(\cdot)$ is a Tent map, $\alpha \in (0, 1)$ is a control parameter, the initial number is $x_0 \in (0, 1)$,

$$x_{i+1} = F_\alpha(x_i) = \begin{cases} x_i/\alpha, & 0 < x_i < \alpha, \\ (1 - x_i)/(1 - \alpha), & \alpha < x_i < 1. \end{cases} \quad (3)$$

In addition, subtle changes of the plain text affect only a small part of cipher text if we just do this diffusion operation for one round. Li Heng-Jian *et al.* did three rounds to ensure the diffusibility.

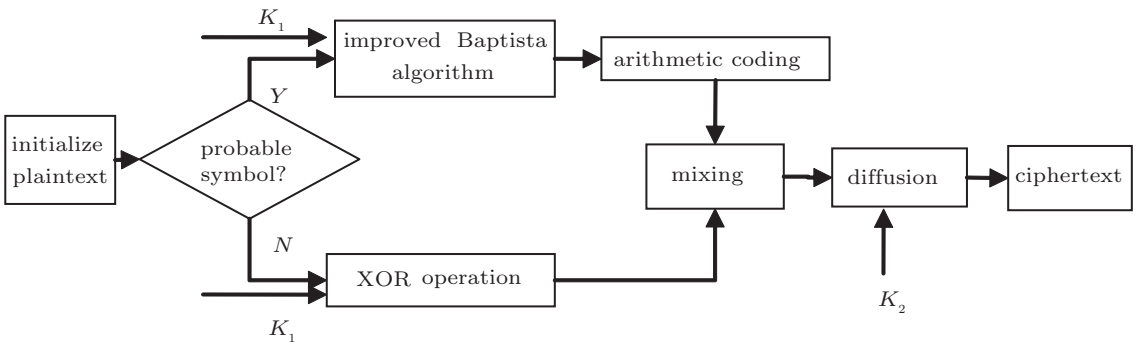


Fig. 2. Encryption flowchart of Li Heng-Jian *et al.*'s algorithm.

2.2. Decryption

(I) Initialize the look-up table as the Step (i) in encryption.

(II) Do the inverted diffusion operation as follows to obtain the mixed sequence S:

$$S_i = f^{-1}(C_i, K_i, \gamma) = ((C_i - AC_{i-1}) \bmod 256) \oplus K_i, \quad (4)$$

where the various parameters are the same as encryption Step (vi).

(III) Separate the mixed sequence S to the compressed sequence and the XOR sequence. Revert the XOR sequence to plain text with the XOR operation and these plain texts are the plain texts of Group 2 by the square.

(IV) Decode the compressed sequence to obtain an iteration sequence and then decrypt the iteration sequence, which is less than MAX, by Baptista algorithm to obtain the plain texts of Group 1. If the number in the iteration sequence is "MAX", iterate the chaos map (i.e., the chaos mapping in the Baptista algorithm, which is commonly selected as logistic map) MAX rounds, and in the corresponding position of plain text of Group 1 insert the plain text of Group 2. If the number in the iteration sequence is "$MAX + 1$", directly insert the plain text of Group 2 in the appropriate position of the plain text of Group 1 without iteration.

3. Cryptanalysis of the Li Heng-Jian *et al.* algorithm

Figure 3 is a diagram of the cryptanalysis of the Li Heng-Jian *et al.* algorithm, which is elaborated below.

3.1. Cryptanalysis of decryption

The decryption steps of the original algorithm only focus on separating the mixed sequence, but, in fact, nobody can accurately separate the mixed sequence S without information about the length of the relevant sequence. If we know neither the length of compressed sequence nor the length of the XOR sequence before separation, we cannot know before which position the two sequences are mixed alternately, and also cannot know after which position only the longer sequence follows.

3.2. Crack the XOR and diffusion part

Which part of plain text is encrypted by XOR and which part of plain text is encrypted by Baptista are decided by the probability of plain text, so we can avoid the Baptista encryption by choosing the low probability plain text. For example, choose single

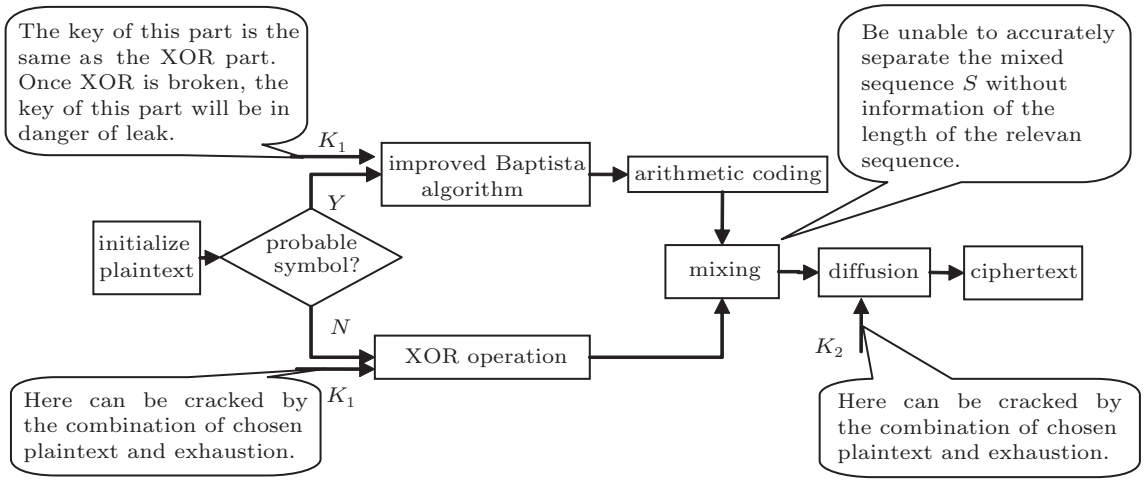


Fig. 3. Sketch map of cryptanalysis.

plain text $P_{1,1} = M + 1$. Because the former M symbol is a high probability symbol when initialized, it is certain that the plain text "$M+1$" will enter into the diffusion part through the XOR part as a low probability symbol. Then the intermediate cipher text sequence from Baptista part will be a single special mark "MAX + 1", and its arithmetic code, i.e. the compressed sequence, is knowable. There is no harm in assuming that the first one of the mixed sequence S is from the XOR sequence and the second one from the Baptista sequence, and then there is the following formula when we enter it into the diffusion equation:

$$\begin{aligned} C_{1,1} &= f(P_{1,1}, K_{1,1}, K_{2,1}, \gamma) \\ &= \{[(((P_{1,1} \oplus K_{1,1} \oplus K_{2,1} + A \cdot \gamma) \bmod 256) \\ &\quad \oplus K_{2,1} + A \cdot \gamma) \bmod 256] \oplus K_{2,1} + A \cdot \gamma\} \\ &\quad \bmod 256, \end{aligned} \tag{5}$$

where the subscript of $C_{i,j}$ and $P_{i,j}$, respectively, denotes the j-th plain text of the i-th time chosen plain text.

Then choose another single plain text $P_{2,1}$ again and let it enter into the diffusion part through the XOR part as mentioned above, such as optionally for $M + 2$.

$$\begin{aligned} C_{2,1} &= f(P_{2,1}, K_{1,1}, K_{2,1}, \gamma) \\ &= \{[(((P_{2,1} \oplus K_{1,1} \oplus K_{2,1} + A \cdot \gamma) \bmod 256) \\ &\quad \oplus K_{2,1} + A \cdot \gamma) \bmod 256] \oplus K_{2,1} + A \cdot \gamma\} \\ &\quad \bmod 256. \end{aligned} \tag{6}$$

Due to

$$(a + b) \bmod n = (a \bmod n + b \bmod n) \bmod n,$$

we can take $A \cdot \gamma$ as a number Q between 0 and 255. Then combine Eqs. (5) and (6), we have

$$\begin{aligned} K_{1,1} \oplus K_{2,1} &= \varphi(C_{1,1}, P_{1,1}, K_{2,1}, Q) \\ &= \{\{[[((C_{1,1} - Q) \bmod 256) \oplus K_{2,1} - Q] \\ &\quad \bmod 256] \oplus K_{2,1} - Q\} \bmod 256\} \\ &\quad \oplus P_{1,1}. \end{aligned} \tag{7}$$

Substitute Eq. (7) into Eq. (6) to eliminate $K_{1,1}$, we have

$$\begin{aligned} C_{2,1} = \{[(((P_{2,1} \oplus \varphi(C_{1,1}, P_{1,1}, K_{2,1}, Q) + Q) \bmod 256) \\ \oplus K_{2,1} + Q) \bmod 256] \oplus K_{2,1} + Q\} \bmod 256. \end{aligned} \tag{8}$$

Here only $K_{2,1}$ and Q are unknown in Eq. (8). They can be obtained by the method of exhaustion and only need 256×256 times exhaustion at most.

It is easy to work out $K_{1,1}$ and γ after working out $K_{2,1}$ and Q. similarly, the subsequent $K_{1,2}, K_{1,3}, \ldots, K_{1,n}$ and $K_{2,2}, K_{2,3}, \ldots, K_{2,n}$ can be cracked by choosing a lower probability plain text. Since the γ has been cracked, the subsequent cracked operation only needs 256 times exhaustion at most every time. Therefore, it only needs $256 \times 256 + 256 \times (n-1)$ times exhaustion to work out the key sequence whose length is n and the number of exhaustion is completely acceptable. Notice that every cracking process is independent, i.e. there is no correlation between the exhaustion of $K_{1,i}$, $K_{2,i}$ and the exhaustion of $K_{1,j}$, $K_{2,j}$ $(i \neq j)$, so the $(n-1)$ here is a multiplicator but not a power.

Sometimes there may be more than one couple of $K_{1,i}$ and $K_{2,j}$, which can meet the conditions to the same $\varphi_{i,j} = K_{1,i} \oplus K_{2,j}$ so that only $\varphi_{i,j}$ can be obtained but cannot respectively confirm the $K_{1,i}$ and $K_{2,j}$ uniquely through the method. However, it does not in fact hinder us cracking the most cipher text

since $\varphi_{i,j}$ is known, because the matching relation of $K_{1,i}$ and $K_{2,j}$ are fixed when the plain text is long.

When every key of K_1 and K_2 must be known uniquely to improve the cracking rate, we can obtain the even position of K_2 because in the diffusion part the even position of K_2 corresponds to the "$MAX + 1$", which is from the Baptista part when all of the chosen plain texts are low probability symbols. Therefore, we can crack the even position of K_2 by first choosing long plain text and then staggering the odd and even positions of the K_1 and K_2 by low probability symbols with various lengths, thereby we can confirm the K_1 and odd position of K_2.

3.3. Cryptanalysis of the Baptista part

Since the key of Baptista part is the same as the XOR part, although the methods of the two parts are different (in that the former makes use of the ergodic property of chaos sequence and the latter makes use of the randomicity of the chaos sequence), once the XOR part had been cracked, the key of the Baptista part would be in danger of being revealed.

Out of consideration for encryption speed, generally speaking, the maximum number of iterations of the Baptista part cannot be too large, for example $MAX = 32$ in Li Heng-Jian *et al.*'s algorithm. Since the plain text whose Baptista iteration number is larger than the maximum iteration MAX will be encrypted by the XOR part instead of by Baptista, the plain text which is processed by the Baptista part will not be much. As the XOR part and the diffusion part are cracked in the former section, most of the plain text in the whole system will be leaked out, which affects the security of the whole system seriously.

4. Improvement of Li Heng-Jian *et al.*'s algorithm

4.1. Use of different keys

Use another key K_3, which also can be selected as the logistic map but the control parameters of K_3 are different from K_1 for the Baptista algorithm of Li Heng-Jian *et al.*'s algorithm.

4.2. Improvement of the XOR part and the diffusion part

As is known, the vulnerability of the Li Heng-Jian *et al.* encryption system is that we can skillfully bypass the Baptista part by choosing plain text and that the virtual key space of the XOR part and the diffusion part are too small. The value range of each key in the two parts is just from 0 to 255. According to that, the improvement measures should increase the range of the keys of the two parts.

Increasing the range of the keys can be done by appending addition, multiplication and other operations, which will cause some problems replating to hardware implementation and encryption efficiency. So we advise changing the XOR part as follows:

$$C_i = LS_{K_{3,i}}(P_i \oplus K_{1,i}), \tag{9}$$

where $LS(\cdot)$ is a ring shift left operation, i.e. ring shift left $P \oplus K_1 \quad K_3$ bits. Due to the appending of the ring shift left operation, it is unfeasible to eliminate $K_{1,i}$ in Subsection 3.2. The length of the binary digit of $P \oplus K_1$ is eight, so the effective value range of $K_{3,i}$ is from 0 to 7. If we still analyze following the ways outlined in Subsection 3.2, it requires $256 \times 256 \times 256 \times 8 = 2^{27}$ times exhaustion and $256 \times 256 \times 8 = 2^{19}$ times subsequent exhaustion, increasing the difficulty of cracking.

Use different keys for the diffusion part each round, i.e.

$$\begin{aligned} C_i = f(f(f(P_i, K_{2,i,1}, C_{i-1}), K_{2,i,2}, C_{i-1}), \\ K_{2,i,3}, C_{i-1}), \end{aligned} \tag{10}$$

under such circumstances, once exhaustion happens, we need $(2^{27})^3 = 2^{81}$ times operations and it is already unable to attack by exhaustion.

4.3. Improvement of mixing part

In order to solve the problem of decryption, we append the difference between the length of the compressed sequence and the length of the XOR sequence, i.e.

$$\begin{aligned} L = &\text{Length(compressed sequence)} \\ &- \text{Length(XOR sequence)} \end{aligned}$$

to the end of the mixed sequence. As the value range of the mixed sequence is from 0 to 255, the value of L should be normalized to that from 0 to 255 in encryption. For instance, express it in the form of a polynomial:

$$L = L_{n-1} \times 256^{n-1} + L_{n-2} \times 256^{n-2} + \cdots + L_0,$$

in which the value range of each polynomial coefficient L_i is from 0 to 255. Append the polynomial coefficients $L_0 \sim L_{n-1}$ to the tail of the mixed sequence in turn. Finally, append the number of the coefficient n

to the tail of the mixed sequence in order to revert the L in decryption.

During the process of decryption, if $L > 0$ then cut the last L number of mixed sequence to make them the last L number of the compressed sequence, if $L < 0$ then cut the last L number of the mixed sequence to make them the last L number of the XOR sequence, and then alternately collect the compressed sequence and the XOR sequence from the left of the mixed sequence. If $L = 0$ then alternately collect the compressed sequence and the XOR sequence from the mixed sequences without cutting.

The purpose of keeping the difference between the two sequences instead of respectively keeping the length of the two sequences is to hide information about the two sequences better and to more effectively economize memory space.

5. Test and experiment

5.1. Test of cracking

We did this crack test by the first 100 bytes of standard file paper1 of the Calgary Corpus (ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus). Suppose M= 16, the control parameter of the logistic map used in the XOR part and the Baptista part is $K_1 = (x_0,\ b) = (0.4,\ 3.76)$, the control parameter of the Tent map used in the diffusion part is $K_2 = (x_0,\ \alpha) = (0.11,\ 0.4)$, the phase space number is $N = 256$, the maximum of iteration is MAX = 32, the update period is period = 64, the diffusion factor is $A = 1$ and $C_0 = \gamma = 3$. There are 82 K_1 keys and 133 K_2 keys used in encryption.

During the process of cracking, firstly work out the even position of K_2 through 800 chosen plain texts. Secondly work out 78 K_1 keys and 129 K_2 keys through the chosen plain texts whose lengths are from 1 to 6, 12 to 30, 40, 41, 100 and 101, respectively. Lastly, use these cracked keys to revert the cipher text back to plain text. The experimental results show us that 100 bytes of cipher text can work out as 79 bytes of plain text in this test and the success rate reaches 79%. What is more, the plain texts which are worked out take on very good readability and we can do further cracking by the use of context. Table 1 below is a correspondence table which shows the relationship between the length of chosen plain text and the length of the mixed sequence while cracking.

Table 1. Length relation between the chosen plain text and the mixed sequence.

Length of chosen plain text	Length of mixed sequence	Length of chosen plain text	Length of mixed sequence	Length of chosen plain text	Length of mixed sequence
1	3	13	11	25	16
2	4	14	11	26	17
3	4	15	12	27	17
4	5	16	12	28	18
5	6	17	13	29	18
6	7	18	13	30	19
7	7	19	14	40	22
8	8	20	14	41	23
9	8	21	15	100	39
10	9	22	15	101	40
11	10	23	16	800	107
12	10	24	16		

Annotation: the length of the mixed sequence expressed by Matlab expression is size(AC(ones(1, L) $\times$ ($MAX + 1$))), where AC means the function of arithmetic code, L means the length of chosen plain text, MAX means the maximum iteration. As all of the chosen plain texts are low probably ones, they are all encrypted through the XOR part and synchronously append the mark "$MAX + 1$" to the Baptista sequence. The Baptista sequence is composed of L "$MAX + 1$" marks and then the compressed sequence is AC(ones(1, L)$\times$($MAX + 1$)).

Table 2 below shows some locations corresponding to the relationship between the key sequence K_1 and K_2 in the diffusion part. From Table 2, it can be seen that while the even position of K_2 is already confirmed, all the positions of K_1 and the odd position of K_2 can also be confirmed by staggering the

odd and even positions of K_1 and K_2. For example, while the length of the chosen plain text is 40, $K_{2,46}$ corresponds to $K_{1,24}$ so that $K_{1,24}$ can be confirmed by $K_{2,46}$. Then, while the length of the chosen plain text is 41, $K_{2,47}$ corresponds to $K_{1,24}$ so that $K_{2,47}$ can be confirmed by $K_{1,24}$.

In addition, the reason why both K_1 and K_2 have four keys cannot be confirmed is that there is no more information while the length of chosen plain text is from 7 to 11. When the length of chosen plain text is from 1 to 6, the chosen plain text is longer than mixed sequence so that the corresponding K_2 can be confirmed by the over part of the mixed sequence. When the length of the chosen plain text is larger than 12, the difference between the mixed sequence and the chosen plain text is larger than or equal to 2 so that K_1 and K_2 can be confirmed by the odd–even staggered method mentioned above with the over part of the chosen plain text. When the length of chosen plain text is from 7 to 11, there is no redundant part of the mixed sequence, and the corresponding positions of K_1 and K_2 is fixed, so we cannot stagger the odd and even positions of K_1 and K_2. Therefore, there are 8 keys i.e. $K_{2,15}$, $K_{2,17}$, $K_{2,19}$, $K_{2,21}$ and $K_{1,8\sim11}$ that cannot be confirmed uniquely. However, when the plain text is so long, the corresponding relation is fixed in the diffusion part so that it will not affect our crack.

Table 2. Location relation of K_1 and K_2.

Length of chosen plain text	Demonstration of the location relation of K_1 and K_2 (where m_i is one of the mixed sequence)									
1	$K_{1,1}$	m_1	m_2	m_3						
	$K_{2,1}$	$K_{2,2}$	$K_{2,3}$	$K_{2,4}$						
2	$K_{1,1}$	m_1	$K_{1,2}$	m_2	m_3	m_4				
	$K_{2,1}$	$K_{2,2}$	$K_{2,3}$	$K_{2,4}$	$K_{2,5}$	$K_{2,6}$				
3	$K_{1,1}$	m_1	$K_{1,2}$	m_2	$K_{1,3}$	m_3	m_4			
	$K_{2,1}$	$K_{2,2}$	$K_{2,3}$	$K_{2,4}$	$K_{2,5}$	$K_{2,6}$	$K_{2,7}$			
4	$K_{1,1}$	m_1	$K_{1,2}$	m_2	$K_{1,3}$	m_3	$K_{1,4}$	m_4	m_5	
	$K_{2,1}$	$K_{2,2}$	$K_{2,3}$	$K_{2,4}$	$K_{2,5}$	$K_{2,6}$	$K_{2,7}$	$K_{2,8}$	$K_{2,9}$	
...										
30	...	m_{19}	$K_{1,20}$	$K_{1,21}$	$K_{1,22}$	$K_{1,23}$	$K_{1,24}$	$K_{1,25}$	...	$K_{1,30}$
	...	$K_{2,38}$	$K_{2,39}$	$K_{2,40}$	$K_{2,41}$	$K_{2,42}$	$K_{2,43}$	$K_{2,44}$	...	$K_{2,49}$
40	...	m_{22}	$K_{1,23}$	$K_{1,24}$	$K_{1,25}$	$K_{1,26}$	$K_{1,27}$	$K_{1,28}$	...	$K_{1,40}$
	...	$K_{2,44}$	$K_{2,45}$	$K_{2,46}$	$K_{2,47}$	$K_{2,48}$	$K_{2,49}$	$K_{2,50}$	...	$K_{2,62}$
41	...	m_{23}	$K_{1,24}$	$K_{1,25}$	$K_{1,26}$	$K_{1,27}$	$K_{1,28}$	$K_{1,29}$	...	$K_{1,41}$
	...	$K_{2,46}$	$K_{2,47}$	$K_{2,48}$	$K_{2,49}$	$K_{2,50}$	$K_{2,51}$	$K_{2,52}$	...	$K_{2,64}$
100	...	m_{39}	$K_{1,40}$	$K_{1,41}$	$K_{1,42}$	$K_{1,42}$	$K_{1,43}$	$K_{1,44}$	...	$K_{1,100}$
	...	$K_{2,78}$	$K_{2,79}$	$K_{2,80}$	$K_{2,81}$	$K_{2,82}$	$K_{2,83}$	$K_{2,84}$	...	$K_{2,139}$
101	...	m_{40}	$K_{1,41}$	$K_{1,42}$	$K_{1,43}$	$K_{1,44}$	$K_{1,45}$	$K_{1,46}$	...	$K_{1,101}$
	...	$K_{2,80}$	$K_{2,81}$	$K_{2,82}$	$K_{2,83}$	$K_{2,84}$	$K_{2,85}$	$K_{2,86}$	...	$K_{2,141}$

5.2. Experiment with the improved algorithm

The improved algorithm mentioned in this part is improved from the original Li Heng-Jian *et al.* algorithm by the method mentioned in Section 4. As we cannot decrypt without information about the length of the XOR sequence, in the original algorithm used here we append the length of the XOR sequence to the tail of the mixed sequence. Although we add some operations in the improved algorithm for security, experiments show that the compression ratio of the improved algorithm does not decline and the encryption/decryption time of the improved one is in the same order of magnitude of the original one. Figure 4 shows that the cipher text value of the improved algorithm is distributed uniformly. Figure 5 shows that the sensitivity of the improved algorithm is high.

Table 3 below is a compression ratio comparison of the original algorithm and the improved algorithm, where the control parameters are same as those in the

Subsection 5.1, and the key of the logistic map in the Baptista part is $K_3 = (x_0, b) = (0.4, 3.7)$.

From Table 3 we can see that the compression rates of the two algorithms are approximately the same. The improved one has a slightly lower compression rate than the original one with the experimental parameters used here.

Table 4 is the encryption/decryption comparison of time between the original algorithm and improved algorithm. Table 4 shows us that the decryption speed is a little slower than the encryption in both two algorithms. Due to operations such as joining of left shift, etc. in the improved one, the speed of encryption/decryption of the improved one is little slower than the original one. However, their cost time is almost in the same order of magnitude, so the speed is acceptable.

Table 3. The compression ratio comparison of the original algorithm and the improved algorithm.

Sample	Original algorithm	Improved algorithm	Sample	Original algorithm	Improved algorithm
paper 1	0.9549	0.9484	progc	0.9656	0.9624
paper 2	0.9188	0.9143	progp	0.9148	0.9106
paper 3	0.9265	0.9248	progl	0.9095	0.9036
paper 4	0.9368	0.9341	geo	0.9026	0.8921
paper 5	0.9637	0.9580	pic	0.2055	0.2029
paper 6	0.9612	0.9498	bib	0.9781	0.9685
News	0.9711	0.9547	trans	0.9963	0.9904
book 1	0.9142	0.9032	obj1	0.9099	0.9033
book 2	0.9291	0.9272	obj2	0.9942	0.9920

Annotation: sample is from ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus.

Table 4. The encryption/decryption comparison of time between the original algorithm and the improved algorithm.

Encryption/s			Decryption/s		
Sample	Original algorithm	Improved algorithm	Sample	Original algorithm	Improved algorithm
paper1	0.156	0.171	progc	0.171	0.203
paper2	0.218	0.250	progp	0.265	0.296
paper3	0.140	0.156	progl	0.172	0.281
paper4	0.062	0.063	geo	0.078	0.0778
paper5	0.062	0.062	pic	0.078	0.078
paper6	0.125	0.140	bib	0.156	0.156
news	0.843	0.906	trans	0.969	1.078
book1	1.640	1.765	obj1	1.968	2.156
book2	1.344	1.453	obj2	1.609	1.735
progc	0.125	0.125	progc	0.140	0.156
progp	0.141	0.157	progp	0.156	0.172
progl	0.203	0.219	progl	0.234	0.234
geo	0.359	0.375	geo	0.391	0.422
pic	1.531	1.578	pic	1.844	1.875
bib	0.266	0.297	bib	0.312	0.359
trans	0.235	0.251	trans	0.266	0.282
obj1	0.109	0.109	obj1	0.110	0.125
obj2	0.875	0.922	obj2	0.969	1.031

Annotation: sample is from ftp://ftp.cpsc.ucalgary.ca /pub/projects/text.compression.corpus.

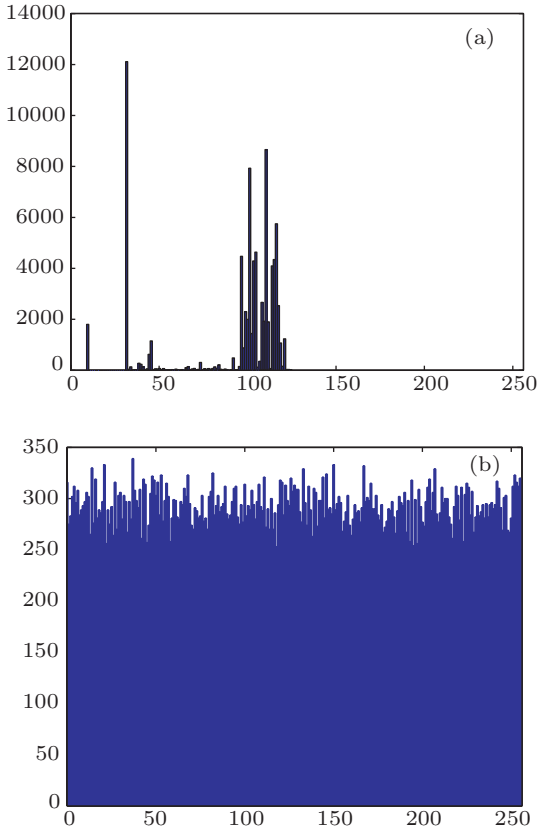


Fig. 4. The plain text value and the cipher text value distribution of the same plain text, (a) the plain text value distribution and (b) the cipher text value distribution. Annotation: the sample is the first 100 values of paper2 from the following web site: ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus.

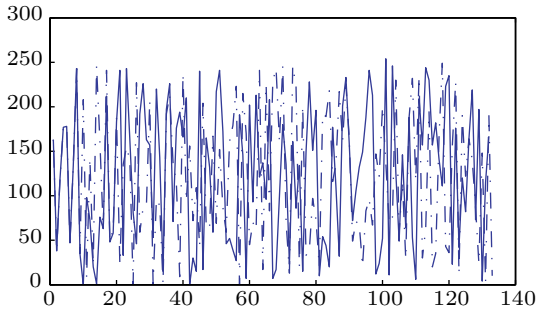


Fig. 5. The result of the sensitivity analysis of the secret key. Annotation: the sample is the first 100 values of paper2 from the following web site: ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus. The secret key analysed here is K_3, whose control parameter b charges 0.001.

Figure 4 is the plain text value and the cipher text value distribution of paper 2, which is mentioned in Table 3 and Table 4. Figure 4 shows that the cipher text value is distributed uniformly.

Figure 5 is the result of sensitivity analysis of the secret key. As shown in Fig. 5, if the secret key changes a little, the cipher text will vary obviously, which means that the improved algorithm has high sensitivity when the secret key changes.

6. Conclusion

This paper analysed a chaos-based cryptosystem with an embedded adaptive arithmetic coder, proposed by Li Heng-Jian *et al*, and then pointed out a decryption bug and a security loophole in the original algorithm. After that, this paper proved and explained a possible cracking method of the XOR part and the diffusion part in detail. In response to the bug and loophole of the original algorithm, this paper also proposed some feasible suggestions for improvement and then proved the validity of the crack and the feasibility of improvement by test and experiment.

References

[1] Baptista M S 1998 *Phys. Lett.* A **240** 50
[2] Wang K, Pei W J, Zou L H and He Z Y 2006 *Acta Phys. Sin.* **55** 6243 (in Chinese)
[3] Long M, Peng F and Chen G R 2008 *Chin. Phys.* B **17** 3588
[4] Zhang H G, Ma T D, Yu W and Fu J 2008 *Chin. Phys.* B **17** 3616
[5] Jin J X and Qiu S S 2010 *Acta Phys. Sin.* **59** 792 (in Chinese)
[6] Alvarez G, Montoya F, Romera M and Pastor G 2003 *Phys. Lett.* A **311** 172
[7] Alvarez G, Montoya F, Romera M and Pastor G 2004 *Phys. Lett.* A **326** 211
[8] Wong W K, Lee L P and Wong K W 2001 *Comput. Phys. Commun.* **138** 234
[9] Wong K W 2002 *Phys. Lett.* A **298** 238
[10] Wong K W 2003 *Phys. Lett.* A **307** 292
[11] Li S, Chen G, Wong K W, Mou X and Cai Y 2004 *Phys. Lett.* A **332** 368
[12] Wang X Y, Duan C F and Gu N N 2008 *Int. J. Mod. Phys.* B **22** 901
[13] Bose R and Pathak S 2006 *IEEE Trans. CAS-I* **53** 848
[14] Mi B, Liao X and Chen Y 2008 *Chaos, Solitons and Fractals* **38** 1523
[15] Lian S, Sun J, Wang J and Wang Z 2007 *Chaos, Solitons and Fractals* **34** 851
[16] Li H J and Zhang J S 2010 *Chin. Phys.* B **19** 050508
[17] Wong K W and Yuen C H 2008 *IEEE Trans. CAS-II* **55** 1193
[18] Witten I H, Neal R M and Cleary J G 1987 *Commun. ACM* **30** 520
[19] Li H J and Zhang J S 2009 *Commun. Nonlinear Sci. Numer. Simul.* **14** 4304
[20] Wong K W, Lin Q Z and Chen J Y 2010 *IEEE Trans. CAS-II* **57** 146