Embedding adaptive arithmetic coder in chaos-based cryptography*

Li Heng-Jian(李恒建)† and Zhang Jia-Shu(张家树)‡

Sichuan Province Key Lab of Signal & Information Processing, Southwest Jiaotong University, Chengdu 610031, *China*

(Received 23 August 2009; revised manuscript received 19 November 2009)

In this study an adaptive arithmetic coder is embedded in the Baptista-type chaotic cryptosystem for implementing secure data compression. To build the multiple lookup tables of secure data compression, the phase space of chaos map with a uniform distribution in the search mode is divided non-uniformly according to the dynamic probability estimation of plaintext symbols. As a result, more probable symbols are selected according to the local statistical characters of plaintext and the required number of iterations is small since the more probable symbols have a higher chance to be visited by the chaotic search trajectory. By exploiting non-uniformity in the probabilities under which a number of iteration to be coded takes on its possible values, the compression capability is achieved by adaptive arithmetic code. Therefore, the system offers both compression and security. Compared with original arithmetic coding, simulation results on Calgary Corpus files show that the proposed scheme suffers from a reduction in compression performance less than 12% and is not susceptible to previously carried out attacks on arithmetic coding algorithms.

Keywords: chaos, cryptography, compression, arithmetic coding

PACC: 0545

1. Introduction

Chaotic systems have many interesting properties such as sensitivity to initial values and control parameters, ergodicity and mixing, which are analogous to the confusion and diffusion properties of a good cryptosystem. Many cryptosystems based on chaos have been proposed in the past few years.[1−4]

Using the ergodic property of chaos, a typical chaotic cryptosystem is proposed by Baptista.[5] Each equal-width partitioning interval of chaotic orbits maps to a possible plaintext symbol uniquely and the obtained integer number of iteration is the ciphertext. Motivated by the interest arisen by chaotic cryptosystems, several new algorithms based on variations of Baptista's have been addressed in the past years.[6−8] Some attacks have been reported as tools of breaking the original Baptista-type cryptosystem and its modified versions.[9,10] To solve the vulnerability points, some remedial operations are made in Refs. [11]–[13]. To ensure the feasibility and the security, the piecewise linear chaotic map, which has a uniform invariant density function is employed to design dynamical look-up table in Ref. [6]. One drawback of the Baptista-type chaotic cryptosystem is that the length of ciphertext in Ref. [5] is at least twice that of plaintext. Though some variant versions are proposed to reduce the length of ciphertext,[7,11] the ciphertext cannot be shorter than the plaintext for the type of cryptosystem. This property of Baptista-type cryptosystem leads to huge ciphertext files and it is not suitable for transmitting and storage especially for large multimedia files.

Entropy coding that provides encryption while still preserves compression property has been developed during recently years.[14−17] However, these entropy-based cryptographic schemes are not secure enough. For example, the encryption in multiple Huffman tables and arithmetic coding with key-based interval splitting are vulnerable to known plaintext attacks.[18] In light of adaptive modeling which offers a huge model varying in structure, a chaos-based adaptive arithmetic coding-encryption technique is designed in Ref. [16]. However, cryptanalysis reveals that this scheme cannot resist a chosen-plaintext attack.[19] Considering that the ciphertext in chaos-based cryptosystem is not balanced (not uniform distributed), Wong and Yuen[20] incorporated Huffman

*Project supported by the National Natural Science Foundation of China (Grant No. 60971104), the Basic Research Foundation of Sichuan Province, China (Grant No. 2006J013-011), and the Outstanding Young Researchers Foundation of Sichuan Province, China (Grant No. 09ZQ026-091).

†E-mail: hengjianli2000@126.com

‡Corresponding author. E-mail: jszhang@home.swjtu.edu.cn

© 2010 Chinese Physical Society and IOP Publishing Ltd

coder into cryptographic schemes to reduce the length of cihertext while enhancing the security. However, the performance has lower compression efficiency. To improve the performance, adaptive arithmetic coding, which is utilized in the novel image/video generation standards (JPEG2000 and H.264), is proposed to compress the number of iterations in this study. What is more, the more portable symbols are adaptively determined which is usually required during the practical realizations. And in order to achieve the maximum compression ratio, the phase space is divided nonuniformly according to the symbol probability. The lookup tables employed in the search mode are updated adaptively by the probability of plaintext symbols. As a result, more portable symbols will have more chances to be visited by the chaotic search trajectory. The required number of iterations is thus small and so can be represented by a short code. Also, to further improve the compression efficiency, the piecewise linear chaotic map (PWLCM) which has uniform distribution is employed in the search mode chaotic cryptosystem. And also the design of dynamical look-up table is to ensure the security. Simulations verify that the proposed scheme can achieve both compression and encryption.

The rest of this paper is organized as follows. The Baptista-type chaotic cryptosystem is reviewed in Section 2. Section 3 describes the embedding adaptive arithmetic coder in the chaos-based cryptosystem. Simulation results and security analyses can be found in Section 4. Conclusions are drawn in the last section.

2. The Baptista-type chaotic cryptosystem

A typical chaos-based cryptographic scheme was proposed by Baptista.[5] The chaotic map chosen in the scheme is the logistic map

$$X_{i+1} = bX_i\left(1 - X_i\right), \tag{1}$$

where $b \in [3.57, 4]$, $X_i \in (0, 1)$ and $i = 0, 1, ...$ A paragraph of plaintexts is divided into a number of message blocks. The plaintext can be represented by the ASCII code of the character. Accordingly, an interval $X' = [x_{\min}, x_{\max}) \subseteq X$ (usually the interval is (0.2 0.8)) is divided into N (=256) equal-width sites. Each site is associated with a plaintext ASCII code. Let $\varepsilon = (x_{\max} - x_{\min})/N$ be the width of each sites, S_i be the site in the range $[x_{\min} + \mathrm{i}\,\varepsilon, x_{\min} + (i + 1)\,\varepsilon)$ and a_i is the corresponding plaintext. The above association can be represented as follows:

$$\begin{aligned} H : S &= \{S_0, S_1, \ldots, S_{N-1}\} \\ \rightarrow A &= \{a_0, a_1, \ldots, a_{N-1}\}. \end{aligned} \tag{2}$$

The plaintext is encrypted as the number of iterations applied in the chaotic map. A secret chaotic trajectory generated from the key-dependence chaotic map parameter and initial condition is utilized to search for the partition mapped to the plaintext symbol being encrypted. The length of the searching trajectory is equal to the number of iterations of the logistic map, which is then taken as the ciphertext. In other words, the partitioned phase space together with the corresponding plaintext mapping can be considered as lookup table or a codebook for encryption. In the decryption process, the same secret plaintext symbols are recovered only if the same secret key and the lookup table are available.

In the modified Baptista-type chaotic cryptosystem,[20] the symbols do not have the uniform partitions in the chaotic phase but the more probable symbols have more partitions. As a result, the required number of iterations for more probable symbols becomes small while that of the others gets large. According to information theory, the compression ability is achieved by exploiting the nonuniformity in the probabilities. As an example, suppose that there are only five possible plaintext symbols ('A', 'B', 'C', 'D', 'E') with probabilities of occurrence 6/16, 4/16, 3/16, 2/16 and 1/16, respectively. If the phase space [0,1] is divided into 256 partitions, then 96 of them should map to symbol 'A', 64 to symbol 'B', 48 to symbol 'C', 32 to symbol 'D', 16 to symbol 'E', as shown in Fig. 1. It should be noticed that the partitions mapped to the symbol are not necessarily at adjacent positions. The phase space of the chaotic map is divided into a number of equal-width partitions, and the number of partitions mapped to a particular symbol is proportional to its probability of occurrence. While the number of iterations for more probable symbols is small, it can be very large for less probable symbols since there are only one or a few partitions mapped to them. By exploiting non-uniformity in the probabilities, the compression capability is achieved.

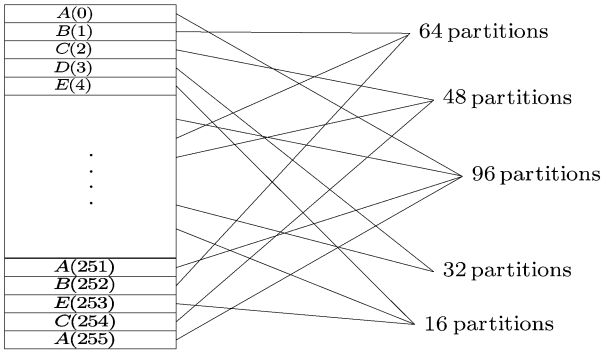


Fig. 1. The look-up table. Mapping of the plaintext symbols to partitions in the orbit of the chaotic map. The total partitions is 256 and there is only five plaintext symbols (A, B, C, D, E) with probabilities of occurrence 6/16, 4/16, 3/16, 2/16 and 1/16.

3. The proposed embedding compression cryptosystem

Compared with the original Baptista-type chaotic cryptosystem, the main difference in Ref. [20] is that the more probable symbols are chosen according to the occurrence of each plaintext symbol. The whole plaintext is scanned firstly to obtain the symbols and their corresponding lookup tables. However, practical realizations generally require running adaptive estimation of these probable symbols because sufficient statistical knowledge of the plaintext is lacking or because its statistics are time varying, correspondingly, the look-up tables should be updated dynamically. The estimation of the probabilities of the source symbols during the encryption process is needed. There are several strategies for computing the occurrence of each plaintext symbol. (a) Use a constant distribution that is available, normally estimated by the gathering statistics in a large number of typical samples. (b) Use predefined distributions with adaptive parameter estimation. For instance, assume that the data has Gaussian distribution and only estimate the mean and variance of each symbol. (c) Use a distribution based on the occurrence of symbols previously appeared. The methods (a) and (b) are usually for special data source. For general data source, the strategy (c) is usually employed. Once the probabilities have been estimated, the more symbols and their lookup tables can be determined easily.

After the number of iterations in the Baptista-type chaotic cryptosystem is obtained, the non-uniform of iterations is compressed by arithmetic coding. Different from the block code (the output of Huffman coding), we use adaptive arithmetic coding which replaces the source ensemble by a code string to compress the number of iterations. The arithmetic coding achieves an optimum which exactly corresponds to the theoretical specifications of the information theory. A slight degradation results from inaccuracies, which are caused by correction mechanisms for the interval division. In contrast to this, the Huffman coding always produces rounding errors, because its code length is restricted to multiples of a bit. This deviation from the theoretical optimum is much higher in comparison to the inaccuracies of arithmetic coding. The efficiency of an arithmetic code is always better or at least identical to a Huffman code. To guarantee the compression property, not all the plaintext symbols should be included in the mapping table, and only a limited number of more probable ones can be selected. Less probable symbols not found in the mapping table are encrypted by encrypting them with a keystream also generated by the chaotic map. Extra special symbols are required to distinguish between the plaintext symbols that are encrypted in that mode.

3.1. Encryption

The encryption procedure of the proposed scheme is illustrated in Fig. 2, and described in detail as follows.

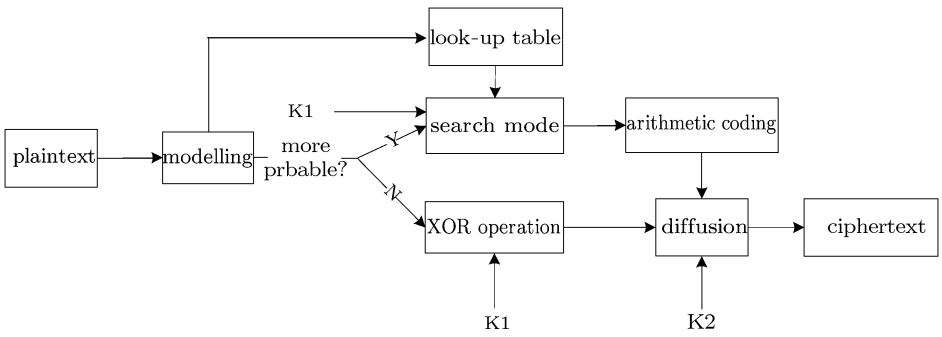


Fig. 2. Encryption block diagram of the proposed cryptosystem.

Step 1) Initially all the number of symbol might be the same(reflecting no initial information). Then, Select the most M probable symbols (such as the first M symbols).

Step 2) Building the lookup tables according to the probable M symbols and their probabilities dynamically. Firstly, divide the phase space into N equal-width partitions. Then, starting from $j = 1$, map the M symbols to the N partitions using the same strategy in Ref. [20] as followings.

$$n(s_j) = \left\lfloor \frac{N \times u(s_j)}{\sum_{i=1}^{M} u(s_j)} \right\rfloor + 1, \tag{3}$$

where $u(s_j)$ and $n(s_j)$ are the number of occurrence and the number of partitions mapped to symbol s_j, respectively. There may be the case that all the partitions are mapped and nothing is left for the last few portable plaintext symbols. However, this does not matter for the more probable symbols should have higher priorities. The symbol is encrypted as the number of iterations applied in the lookup tables.

Step 3) Divide the plaintext being processed into two groups. One is the M selected symbols while the left symbols are for the other group. For the M selected symbols, iterate the chaotic map T times until the trajectory falls into the any desired partitions which are associated with the symbol. If the iteration times go beyond the maximum number of iterations (call it 'Max') for short which is pre-defined in the search mode, the more probable symbol is denoted as a less probable symbol which will be encrypted in the XOR mode. If the symbol being processed is not the probable, a special symbol (for example, Max+1) is inserted into the iteration numbers. In the same way, we can obtain the ciphertext for the search mode. From the above, we know that the length of the iteration times is equal to that of the plaintext.

Step 4) The number of iterations and the special symbol are compressed via arithmetic coding while the left plaintext symbols are masked via a XOR way. The unit in the output of arithmetic encoding is in Byte. Different from the block code of Huffman coding, adaptive arithmetic coding replaces the source ensemble by a code string. So the following method is made to make the output of arithmetic coding and the XOR-ed plaintext mix together. That is, the arithmetic coding and the XOR-ed ciphertext given the output in turn, for example, one is the Bytes of arithmetic coding, and then the other is the XOR-ed ciphertext. And the remaining are put together in the end. If there is more than one mapping look-up table employed, of course, the number of iterations with different mapping look-up tables should form different Statistical Modeling in arithmetic coding.

Step 5) Update the probability of the symbols which have been appeared and choose the more probable symbols on condition that the number of being processed ciphertext is up to the period. Then goto step 2.

Step 6) After all the plaintext have been processed, the obtained intermediate sequences are implemented with diffusion operation to get the final ciphertext. The diffusion operation is implemented as follows:

$$\begin{aligned} C_i &= f(P_i, K_i, \gamma) \\ &= (T_i \oplus K_i + A \cdot C_{i-1}) \mod 256, \end{aligned} \tag{4}$$

where γ is an integer related to key and the initial cipher $C_0 = \gamma$. P_i is the mixture of number of iterations and XOR-ed plaintext. A is a diffusion factor, and K_i can be obtained by the following formula:

$$\begin{aligned} K_i &= \left(\text{round}\left(F_\alpha(x_{i-1}) \times 2^{31}\right)\right) \mod 256, \\ & i = 1, 2, \ldots, \end{aligned} \tag{5}$$

where $F_\alpha(\cdot)$ is tent chaotic map, $\alpha \in (0, 1)$ is control parameter, the initial value is $x_0 \in (0, 1)$.

$$x_{i+1} = F_\alpha(x_i) = \begin{cases} x_i/\alpha, & 0 < x_i < \alpha, \\ (1 - x_i)/(1 - \alpha), & \alpha < x_i < 1. \end{cases} \tag{6}$$

If this masking process is implemented only once, a tiny change in the last of the plaintext may affect only a small proportion of the ciphertext. Therefore, in our scheme, it is implemented three times consecutively so as to ensure the diffusion.

3.2. Decryption

Unlike the static encryption in Ref. [20], the plaintext-specific information such as the probable plaintext symbols and their number of occurrence do not need to be transmitted since the required information can be reconstructed dynamically in the adaptive model for encryption. That is also one of the advantages in the adaptive mode. An adaptive model represents the changing symbol frequencies seen so far in a message. Initially all counts might be the same (reflecting no initial information), but they are updated, as each symbol is seen, to approximate the observed frequencies. Provided both encryption and decryption

the same initial values (e.g., equal counts) and the same periodic adaptation strategy for choosing more probable symbols, their models will remain the same. The encryption receives the next symbol, encrypts it, and updates its model. The decryption identifies it according to its current model and then updates its model.

The decryption procedure is similar to that of the encryption process mentioned above. Since the chaotic maps are deterministic, the receiver can decrypt message exactly by using the same secret keys. However, the following should be emphasized.

Step 1) Firstly, initialize the look-up-table.

Step 2) Inverse the diffusion operation in the following manner:

$$\begin{aligned} P_i &= f^{-1}(C_i, K_i, \gamma) \\ &= ((C_i - A \cdot C_{i-1}) \mod 256) \oplus K_i. \end{aligned} \quad (7)$$

The initial values and the control parameters are the same as those of step (6) in encryption procedure. Then, we get the string of iteration with arithmetic encoding and the XOR-ed plaintext.

Step 3) Separate the search and XOR mode from the intermediate sequences and use the arithmetic decoding to obtain the number of iterations. For the XOR mode, we can direct obtain the plaintext via XOR again with the same key.

Step 4) Implement a reverse operation described in step 3 in the encryption procedure with the same shared secret parameters and initial values. It needs to iterate the maximum times with the chaotic map on the condition that the symbol is one of the more probable symbols and the number of iteration is up to the maximum. If the number of iteration meets the special symbol 'Max+1', the special symbol will be replaced by the original plaintext. The look-up tables will be updated if the number of decryption symbol is up to an integer multiple of the cycle in encryption.

In this study, the probability estimation with periodic updating is employed for balance between the computation burden and efficiency. We call periodic adaptation the process of updating the probability estimation only after a certain number of symbols has been processed. In this way, the adaptive chosen more probable symbols respond well to local statistical properties. Another advantage of the system is that it is required only one pass over the plaintext. Of course, one-pass methods are not very interesting if the number of bits they transmit is significantly larger than that of the two-pass scheme. Interesting, the performance of the method, in terms of number of bits transmitted, can be better than that of choosing more probable symbols statically.

4. Experiments and discussions

The piecewise linear chaotic map (PWLCM) has a uniform distribution

$$h(w) = \begin{cases} w/p, & 0 \le w < p, \\ (w-p)/(0.5-p), & p \le w < 0.5, \\ (1-w-p)/(0.5-p), & 0.5 \le w < 1-p, \\ (1-w)/p, & 1-p \le w < 1, \end{cases} \quad (8)$$

where p is a control parameter. Since the PWLCM is ergodic and possesses uniform invariant density function, it not only enables the whole attractor serving as the target interval, but also prevents guessing the plaintext. The proposed algorithm is implemented in VC^{++}6.0 platform on a personal computer with XP operating system. The hardware of the computer is configured with an Intel Celeron IV 2.0 GHz processor with 512 MB RAM. The arithmetic coder derives from WNC (Witten-Neal-Cleary) coder.[21] The plaintext symbols are read in bytes and the chaotic trajectory is divided into 256 equal-width partitions. The period of updating Lookup table is 64. The compression and the security analysis will be discussed in the following section. In the experiments, the control parameter $p = 0.35$, the initial value is 0.7109.

4.1. Comparisons between Logistic map and PWLCM

Take paper2.txt as an example, Fig. 3 illustrates the ciphertext of the original Baptista-type chaotic cryptosystem using Logistic map and PWLCM respectively. From the figure, we can see that the number of iteration using the Logistic map is longer than that of PWLCM as a whole. The ciphertext information entropy in the cryptosystem using the Logistic map is 8.5885 while that of PWLCM is 8.3715. That is, substituting the Logistic map with the PWLCM can reduce the volume of ciphertext. The average number of iterations using PWLCM (about 124) is shorter than that of the Logistic map(about 145) since the output of the PWLCM is ergodic and uniformly distributed in the phase space. The experimental results also demonstrate that the using of the PWLCM can

improve the encryption speed.

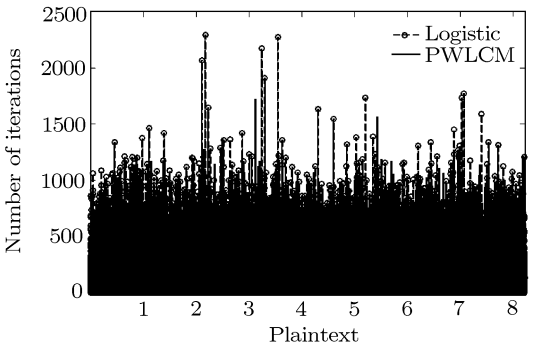


Fig. 3. Ciphertext distribution using the Logistic Map and PWLCM. The control parameter of the Logistic map is 3.999999991 and the initial vale is 0.3388.

4.2. Compression ratio

To test the compression capability of the proposed scheme, the standard files from the Calgary Corpus are downloaded from the website (ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus). There are 18 distinct files of different types, including text, executable, geophysical data, and picture. There are two configurations for experiments. One is that only 16 more probable plaintext symbols are selected and they are all mapped to one table and the maximum number of iterations for the search mode is 32. The other case is that 64 more probable plaintext symbols are selected and they are distributed to 8 tables, each has 8 symbols. The maximum number of iterations in multiple mapping tables is 32. The files are also compressed with arithmetic coding to make a clear comparison between the traditional scheme and the encryption approach.

The compression ratio (ciphertext-to-plaintext) is shown in Table 1 for different files with the two configurations. From Table 1, we can see that the compression performance of the configuration (8 maps, each 8 symbols) is better than the configuration (16 maps, each 8 symbols) of Huffman coding.

Table 1. Comparison between Huffman coding and adaptive arithmetic coding.

file	1 map, 16 symbols		multiple maps, 8 symbols each		entropy coding	
	arithmetic	Huffman	arithmetic (8 maps)	Huffman (16 maps)	arithmetic	Huffman
paper1	0.8426	0.8731	0.7183	0.7923	0.6197	0.6364
paper2	0.8023	0.8423	0.6579	0.7274	0.5753	0.5850
paper3	0.8053	0.8479	0.6756	0.7451	0.5850	0.5956
paper4	0.8098	0.8590	0.6930	0.7731	0.5941	0.6231
paper5	0.8415	0.8813	0.7225	0.8119	0.6244	0.6612
paper6	0.8468	0.8771	0.7217	0.7999	0.6215	0.6431
news	0.8567	0.8803	0.7388	0.8186	0.6460	0.6547
book1	0.7948	0.8339	0.6439	0.7115	0.5662	0.5708
book2	0.8143	0.8509	0.6836	0.7551	0.5950	0.6037
progc	0.8667	0.8931	0.7434	0.8329	0.6505	0.6663
progp	0.8417	0.8609	0.6880	0.7865	0.6082	0.6213
prog1	0.8288	0.8631	0.6789	0.7545	0.5917	0.6062
geo	0.8387	0.8420	0.7084	0.8567	0.7071	0.7212
pic	0.2496	0.3293	0.1306	0.3152	0.1458	0.2092
bib	0.8690	0.8927	0.7544	0.8275	0.6514	0.6578
trans	0.9145	0.9350	0.7988	0.8747	0.6843	0.7015
obj1	0.8870	0.8963	0.7701	0.9169	0.7462	0.8068
obj2	0.9174	0.9295	0.8389	0.9421	0.7589	0.7917

Figure 4 provides a clear compression performance comparison between the proposed encryption and arithmetic coding Fig. 4 shows that our scheme suffers from a decline in compression ratio less than 12% (most of them between 8% and 10%) comparing with arithmetic coding while the performance in Ref. [20] suffers from about 10%–17% comparing with Huffman coding. There are several reasons to explain the experiments. Firstly, the compression performance of arithmetic coding is better than Huffman coding, which is also verified from Table 1. Secondly, the piecewise linear chaotic map with uniform distribution is employed in the search mode, which can improve the compression of number of iterations. Lastly,

we can take the multiple mapping tables as a classification process. In the same mapping table, the symbols have nearly the same probability, which avoid that the more probable symbols pose most of partitions, therefore, there are more chances to visit them and achieve greater compression.

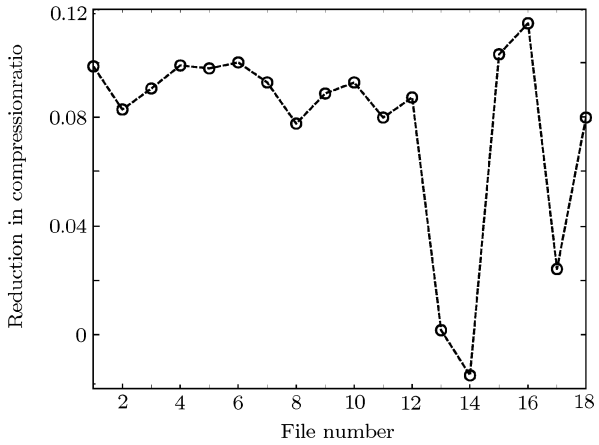


Fig. 4. The compression performance comparison between the proposed encryption and arithmetic coding. The file number is associated with the Table 1, for example, the file number of paper1 is 1, ..., the file number of obj2 is 18.

For the pic.raw file, the information entropy is $H = 1.2102$ bits/symbol, but the average length using arithmetic encoding is 1.1664. We can get a lower average length of symbol, 0.9192 bits/symbol, it we use another arithmetic coding source code(http://www.cipr.rpi.edu/∼said/FastAC.html). The phenomena can be explained in the following way. Our model tacitly assumes that symbols (the whole source) are independent. This is almost not the case. So the true amount of information 'contained' in a sequence of symbols is less than the information entropy. That is, the entropy is computed via adaptive one statistical manner.

4.3. Speed

The encryption and decryption times of implementing the proposed scheme on the Calgary Corpus files for the case of 8 maps, 8 symbols each listed in Table 2. The results show that the speed in the encryption process ranges from 419 KB/s to 618KB/s and that in the decryption process varies from 417 KB/s to 886 KB/s. The implementation time of cascaded operations of Huffman Coding and 128-bit Advanced Encryption Standard (AES) are also given in Table 2 to be compared with the proposed scheme.

Table 2. Encryption and decryption time comparison between Proposed scheme and adaptive arithmetic coding + AES-128.

file	size/byte	encryption (s)		decryption (s)	
		proposed scheme	AC+ AES-128	proposed scheme	AC+ AES-128
paper1	53161	0.1109	0.0917	0.1170	0.0900
paper2	82199	0.1842	0.1280	0.1840	0.1400
paper3	46526	0.1060	0.0688	0.1090	0.0810
paper4	13286	0.0310	0.0215	0.0295	0.0230
paper5	11954	0.0278	0.0198	0.0262	0.0214
paper6	38105	0.0780	0.0590	0.0780	0.0652
news	377109	0.7452	0.5796	0.7765	0.6432
book1	768771	1.7220	1.1328	1.7985	1.3074
book2	610856	1.3392	0.9252	1.3828	1.0340
progc	39611	0.0780	0.0621	0.0810	0.0636
progp	49379	0.0934	0.0735	0.0964	0.0780
prog1	71646	0.1515	0.1045	0.1545	0.1122
pic	513216	0.8249	0.2500	0.5654	0.2855
bib	111261	0.2215	0.1716	0.2324	0.1838
trans	93695	0.1635	0.1420	0.1790	0.1496
geo	102400	0.1590	0.1711	0.1680	0.1790
obj1	21504	0.0340	0.0386	0.0355	0.0415
obj2	246814	0.3985	0.4300	0.4418	0.4680

The speed of our proposed scheme is a little slower than that of the cascaded operations both in encryption and decryption processes except for the last three files (geo, obj1, obj2). The 256-symbol source entropies (bits/symbol) of the files are 5.6464, 5.9482 and 6.2604. The WNC coder is particularly slow at higher source entropies (especially for above 5.5 bits/symbol) because it uses sequential search for coding, which is also very inefficient in the worst case.[23] This is also the reason for that both encryption and decryption in the cascaded operations are much faster than the proposed scheme for the pic.raw file. Compared with the scheme in Ref. [20], the proposed scheme is slow for the following reasons: (1) update the lookup table dynamically; (2) the computation complex of adaptive arithmetic coding is higher than that of static arithmetic coding and Huffman coding;[23] (3) Three diffusion rounds are required in the proposed scheme while the Ref. [20] only needed two rounds. The speed of the proposed scheme could be improved by optimizing the arithmetic coding since the execution efficiency of the WSN Coder is not high.

4.4. Sensitivity analysis

The sensitivity includes the plaintext sensitivity and key sensitivity. To test the sensitivity, we use the mask mode for encryption. For plaintext sensitivity, the changes in the ciphertext should be significant even with a small change in the plaintext. We have tested different positions of the plaintext sequence as shown in the left part of Table 3. The two resultant ciphertext sequences ciphertext sequences are compared bit-by-bit. From the Table 3, the results are all near to 50.00%, which imply that the ciphertext is very sensitive to the plaintext.

Table 3. Sensitivity test.

File	plaintext sensitivity test			key sensitivity test		
	beginning	middle	end	C_0	initial value/10^{-16}	control parameter/10^{-16}
paper1	0.5000	0.5000	0.5000	0.5000	0.4973	0.5009
paper 2	0.5000	0.5000	0.5000	0.5000	0.5020	0.4992
paper 3	0.5000	0.5000	0.5000	0.5000	0.4985	0.4991
paper 4	0.5000	0.5000	0.5001	0.5000	0.5054	0.5018
paper 5	0.5000	0.5001	0.5001	0.5000	0.4930	0.4966
paper 6	0.5000	0.5000	0.5000	0.5000	0.4992	0.4996
News	0.5000	0.5000	0.5000	0.5000	0.5002	0.4989
Book1	0.5000	0.5000	0.5000	0.5000	0.5001	0.5011
Book2	0.5000	0.5000	0.5000	0.5000	0.4998	0.5005
Progc	0.5000	0.5000	0.5000	0.5000	0.4964	0.5005
Progp	0.5000	0.5000	0.5000	0.5000	0.4981	0.5014
Prog1	0.5000	0.5000	0.5000	0.5000	0.4999	0.4983
Geo	0.5000	0.5000	0.5000	0.5000	0.4993	0.4989
Pic	0.5000	0.5000	0.5000	0.5000	0.5005	0.4997
Bib	0.5000	0.5000	0.5000	0.5000	0.5001	0.5015
Trans	0.5000	0.5000	0.5000	0.5000	0.4995	0.5004
Obj1	0.5000	0.5000	0.5000	0.5000	0.5023	0.4985
Obj2	0.5000	0.5000	0.5000	0.5000	0.5001	0.4987

For key sensitivity, two ciphertext sequences are compared and the percentage of bit is computed. For the initial value and the control parameter, the key difference is 10^{-16}, all the percentage data are close to 50%. For C_0, the bit change percentage is 50%. These indicate that the ciphertext is very sensitive to the key.

The sensitivity tests confirm the sensitivity of the ciphertext to the key and the plaintext, even for a bit change at the end of the plaintext. This is because the three rounds of encryption can spread the whole ciphertext. The security of the proposed scheme is high even for the mask mode. There, the hybrid mode is considered more security.

4.5. Key space analysis

A good encryption should resist all kinds of known attacks, it should be sensitive to the secret keys, and the key space should be large enough to make brute-force attacks infeasible. In our algorithm, the initial values of PWCLM map and tent system are used as secret keys, if the precision is 10^{-16}, the key space size is $10^{16\times4} \approx 2^{212}$, about 212 bits. Together with the 8-bit initial cipher C_0, the total key space is 220 bits. This is enough to resist all kinds of brute-force attacks.

4.6. Some other related security problems

There is another two important problems during the implementation of the chaotic map, which also exists in other Baptista-type cryptography. The first one is the weak keys in the whole key space, e.g., the control parameter should not be chosen as 0.25 since it is a weak key. Another efficient solution is that another chaotic map with security properties is replaced, such as feedforward-feedback nonlinear dynamic filter (FFNDF),[22] which does not have weak keys and large key space. However, its computation seems too large if it is implemented in software form. The other one is the finite precision effect. The chaotic map is implemented on finite arithmetic machines, the discrete value realized of a chaotic map shows finite precision effect (a short period cyclic sequence). But in fact, the finite precision effect is not a very serious problem in our encryption algorithm since a dynamic look-up table is generated and the cycles are very high when double precision is adopted. Even though, a perturbation with random sequence is an efficient method to avoid short period.[8]

5. Conclusions

The ciphertext for each message block is the number of iterations required to make the chaotic trajectory fall into the region corresponding to that block and the number could be very large. Therefore, an embedding arithmetic coding compression algorithm in the Baptista-type chaotic cryptosystem is proposed in the paper. The more probable symbols are dynamically selected and an adaptive arithmetic encoder is employed to encode the number of iterations. Therefore, the compression capability is achieved by arithmetic coder (entropy coder), and the system offers compression while the security is still maintained. Compared with original arithmetic coding, the proposed scheme suffers from a reduction in compression performance less than 12%, which is much better than that of those based on Huffman coding.

References

[1] Qiu S S and Xiang F 2008 *Acta Phys. Sin.* **57** 6132 (in Chinese)
[2] Zhou Q, Hu Y and Liao X F 2008 *Acta Phys. Sin.* **57** 5413 (in Chinese)
[3] Xu S J, Wang J Z and Yang S X 2008 *Chin. Phys.* B **17** 4027
[4] Zheng F, Tian X J, Li X Y and Wu B 2008 *Chin. Phys.* B **17** 1685
[5] Baptista M S 1998 *Phys. Lett.* A **240** 50
[6] Wong K W 2002 *Phys. Lett.* A **298** 238
[7] Wong K W 2003 *Phys. Lett.* A **307** 292
[8] Li S J, Chen G R, Wong K W, Mou X Q and Cai Y L 2004 *Phys. Lett.* A **332** 368
[9] Alvarez G, Montoya F, Romera M and Pastor G 2003 *Phys. Lett.* A **311** 172
[10] Alvarez G, Montoya F, Romera M and Pastor G 2004 *Phys. Lett.* A **326** 211
[11] Wong K W, Man K P, Li S J and Liao X F 2005 *Circuits, Systems and Signal Processing* **24** 571
[12] Wei J, Liao X F, Wong K W, Zhou T and Deng Y G 2006 *Phys. Lett.* A **354** 101
[13] Xiao D, Liao X F and Wong K W 2006 *IEEE Trans. Circuits Systems* II **53** 502
[14] Wu C and Kuo C 2005 *IEEE Trans. Multimedia* **7** 828
[15] Wen J, Kim H and Villasenor J D 2006 *IEEE Trans. Signal Process. Lett.* **13** 69
[16] Bose R and Pathak S 2006 *IEEE Trans. Circuits Systems-1* **53** 848
[17] Li H J and Zhang J S 2009 *Commun Nonlinear Sci. Numer. Simulat.* **14** 4304
[18] Jakimoski G and Subbalakshmi K P 2008 *IEEE Trans. Multimedia* **10** 330
[19] Zhou J T and Au O 2008 *IEEE Trans. Circuits Systems* I **55** 3368
[20] Wong K W and Yuen C H 2008 *IEEE Trans. Circuits Systems* II **55** 1193
[21] Witten I H, Neal R and Cleary J G 1987 *Commun. ACM.* **30** 520
[22] Zhang J S, Wang X M and Zhang W F 2007 *Phys. Lett.* A **362** 439
[23] Said A 2004 *Hewlett-Packard Laboratories Report*, HPL–2004–75, Palo Alto, CA. http://www.hpl.hp.com/techreports/2004/HPL-2004-75.pdf