Performing Compression and Encryption Simultaneously using Chaotic Map

Kwok-Wo Wong
Department of Electronic Engineering
City University of Hong Kong
Hong Kong, P.R. China
itkwwong@cityu.edu.hk

Ching-Hung Yuen
Department of Electronic Engineering
City University of Hong Kong
Hong Kong, P.R. China
chyuen@cityu.edu.hk

***Abstract*—An algorithm for performing compression and encryption simultaneously using a chaotic map is proposed. The look-up table used for encryption is determined adaptively by the probability of occurrence of plaintext symbols. As a result, more probable symbols will have a higher chance to be visited by the chaotic search trajectory. The required number of iterations is small and can be represented by a short code. The compression capability is thus achieved. Simulation results show that the compression performance on standard test data and image files is satisfactory while the security is not compromised. Therefore, our scheme can be applied for lossless data and lossy image compression.**

Keywords— chaos, cryptography, compression, encoding

I. INTRODUCTION

In recent years, the size of multimedia files as well as the need for the secure transmission of confidential data over public networks keeps rising. This leads to a growing interest in the investigation of joint operation of compression and encryption on the same set of data, i.e., perform these two operations simultaneously, instead of separately. There are in general two different research directions in this area, to embed encryption into compression algorithms or to incorporate compression in cryptographic schemes.

It is natural to embed encryption in entropy coding such as arithmetic coding and Huffman coding since both cryptographic ciphers and entropy coders bear certain resemblance in the sense of secrecy. An entropy coder is easy to be turned to a cipher by using a secret key to govern the statistical model used in coding. The decoder can track the changes in the statistical model and produces the correct output only if the secret key is available.

The work of embedding encryption into arithmetic coding mainly focused on the control of interval allocation using a secret key. In [1], an arithmetic coding scheme with the functions of encryption, data compression and error detection was proposed. Another scheme utilizes a randomized arithmetic coding paradigm based on key-controlled interval swapping [2]. There was also a suggestion to combine binary arithmetic coding with encryption [3]. By incorporating permutation at the input and output of the encoder, the same research group has recently proposed a modified arithmetic coding scheme with a higher security but negligible coding efficiency penalty [4]. Besides arithmetic coding, other entropy coding schemes such as Huffman coding can also be used for embedding encryption. In [5], a scheme based on multiple Huffman tables was proposed. However, it was lately found that this approach suffers from the chosen-plaintext attack if the Huffman tables are not selected properly [6].

There are some reports on the use of chaos in the joint operation of compression and encryption. In [7], a chaos-based adaptive arithmetic coding technique was proposed. The arithmetic coder's statistical model is made varying in nature according to a pseudo-random bitstream generated by coupled chaotic systems. A similar idea was reported in [8] that the mapping intervals of the arithmetic coder are changed irregularly using a keystream generated from both the chaotic logistic map and the plaintext. A chaotic stream cipher for the selective encryption of video streams was proposed recently [9]. Among all the data in the video stream, only the encoded discrete cosine transform (DCT) coefficients and the sign of the motion vector are encrypted by masking them with a pseudo-random bit sequence generated by two piecewise linear chaotic maps. While all these papers [7-9] utilize chaos in their schemes, they are actually not based on the architecture of chaotic systems, but rather built on the framework of compression such as entropy coding or transform coding. Chaotic systems only play the role of pseudo-random bitstream generators there.

In this paper, an approach for embedding compression in the Baptista-type chaotic cryptosystem [10] is proposed. It roots on the architecture of chaotic cryptosystems rather than the compression framework. The look-up table used for encryption is determined adaptively by the probability of occurrence of plaintext symbols. As a result, more probable symbols will have a higher chance 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. Simulation results verify that the proposed scheme can compress standard test files to a satisfactory degree while performing the encryption. Moreover, our scheme ensures that the ciphertext is not longer than the plaintext.

The rest of this paper is organized as follows. In next section, the Baptista-type chaotic cryptosystem is reviewed. Our approach for embedding compression in this class of chaotic cryptosystem is described in Section III. Simulation results and analyses can be found in Section IV. In the last section, conclusions will be drawn.

II. BAPTISTA-TYPE CHAOTIC CRYPTOSYSTEM

Simple one-dimensional chaotic maps such as the logistic map and the tent map are usually employed for data and document encryption. A typical chaos-based cryptographic scheme was proposed by Baptista [10]. The phase space of

the logistic map is divided into a number of equal-width partitions, each maps to a possible plaintext symbol uniquely. A secret chaotic trajectory generated from the key-dependent chaotic map parameters 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 a look-up table or a codebook for encryption. In the decryption process, the same secret chaotic search trajectory is re-generated and the correct plaintext symbols are recovered only if the same secret key and the look-up table are available.

There are some variants of the Baptista-type chaotic cryptosystem. Wong *et al* modified it to obtain a flatter ciphertext distribution [11]. A dynamic look-up table version was proposed so that the mapping between the phase space partitions and the plaintext symbols keeps changing for different plaintext blocks [12]. As the look-up table is updated dynamically according to the order of appearance of plaintext symbols, it is plaintext dependent and can be considered as a hash or message authentication code (MAC) of the plaintext sequence [13]. In [14], the Baptista-type chaotic cryptographic scheme was further analyzed in detail.

Like other cryptographic schemes, attempts to crack the original Baptista-type chaotic cryptosystem and its variants were made [15, 16]. The causes of vulnerability were investigated and remedial operations were suggested [17-19].

The Baptista-type chaotic cryptosystem suffers from the problem of long ciphertext which is usually about 1.5 to twice the plaintext length [10, 17]. A short-ciphertext variant was suggested in [20] so that the ciphertext is only slightly longer than the plaintext by a short header. Nevertheless, it is believed that the ciphertext cannot be shorter than the plaintext for this type of cryptosystems. In the next section, an algorithm for incorporating compression in Baptista-type chaotic cryptosystems is proposed for lossless and lossy compression.

III. The Proposed Approach

The proposed approach of performing compression and encryption simultaneously can be applied in three different areas, namely, lossless data compression using zero-order and first-order plaintext entropy, and also lossy image compression.

A. Lossless Data Compression using Zero-order Entropy

In this approach, the dynamic look-up table is built adaptively using the plaintext zero-order entropy. By doing so, certain compression capability is achieved along with encryption while the security is not compromised.

First, the whole plaintext sequence is scanned once to find out the probability of occurrence of each plaintext symbol. The phase space of the chaotic map is divided into a number of fixed-width partitions and the number of partitions mapped to a particular symbol is proportional to its probability of occurrence. As an example, suppose that there are only 4 possible plaintext symbols (A, B, C, D) with probability of occurrence 0.5, 0.25, 0.125, and 0.125, respectively. If the phase space (0,1) is divided into 256 partitions, then 128 of them should map to symbol A, 64 to symbol B, 32 to symbol C, and finally 32 to symbol D, as shown in Fig. 1. It should be noticed that the partitions mapped to the same symbol are not necessarily at adjacent positions.

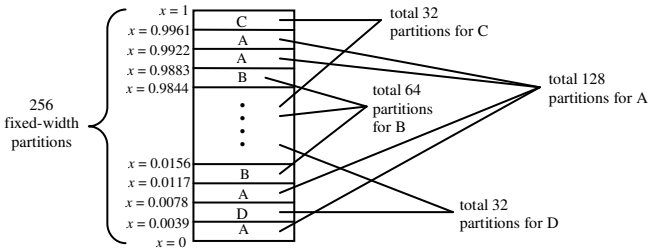

Fig. 1 Mapping of plaintext symbols to partitions in the phase space of the chaotic map.

The advantage of having more partitions for more probable symbols is that the chance for the chaotic trajectory to land on those partitions is also higher. As a result, the required number of iterations is smaller and fewer bits are used to encode it. For example, if each plaintext symbol is represented by a byte and the chaotic search trajectory usually lands on the target partition corresponding to that symbol within 16 iterations. Then a maximum of 4 bits are required for encoding and the ciphertext for that symbol is equal to or shorter than half of the plaintext symbol length. This leads to the compression capability.

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. Therefore we cannot include all the plaintext symbols in the mapping table, but can only choose a limited number of more probable ones. Less probable symbols not found in the mapping table are encrypted by masking them with a pseudo-random bitstream also generated by the chaotic map. As a result, the proposed compression and encryption scheme can be considered as a hybrid one: more probable symbols are encrypted by searching in the dynamic look-up table like block ciphers while less probable ones are masked by a pseudo-random bitstream as performed in stream ciphers. A special symbol is required to distinguish between these two modes and it causes certain overhead to the ciphertext. If this overhead exceeds the gain from encrypting more probable symbols, the ciphertext will become longer than the plaintext. In this case, the whole plaintext sequence will be encrypted solely by masking so as to ensure that the ciphertext is not longer than the plaintext.

B. Lossless Data Compression using First-order Entropy

Besides using the zero-order entropy of the plaintext, the first-order one can also be utilized. This means that the a priori information of the relationship between consecutive plaintext symbols is used to build the look-up table adaptively. By doing so, the chance for the chaotic trajectory to fall into the target partitions increases. The number of iterations required is then small and can be encoded by a short code.

Suppose that there are a total of S possible symbols in the plaintext sequence. For each symbol, scan the whole plaintext sequence to find out all the symbols immediately after it. Then sort them in descending order according to their number of occurrence and select the top K symbols only. As a result, each symbol s_i has its own set $N_i = \{n_{i1}, n_{i2}, n_{i3}, \ldots, n_{iK}\}$ of more probable next symbols, where i=1, 2, …, S. The storage requirement could be very large if the number of occurrence of these K more probable next symbols is stored for each symbol. Therefore the average is taken instead so that all the symbols share the same model of probability of occurrence of next symbols, $P=\{p_1, p_2, \ldots p_K\}$ where p_j is a byte value given by Eq. (1).

$$p_j = \left\lfloor \frac{\sum_{i=1}^{S} u(n_{ij})}{\sum_{i=1}^{S}\sum_{j=1}^{K} u(n_{ij})} \times 256 \right\rfloor + 1 \quad (1)$$

where $u(n_{ij})$ is the number of occurrence of next symbol n_{ij}.

Encrypt the plaintext sequentially by finding the previous symbol and then use the corresponding set N_i. For the first plaintext symbol, there is no previous symbol and so we arbitrarily choose it as 0. Divide the phase space of the chaotic map into M equal-width partitions, with $M > K$. Starting from n_{i1}, map the K more probable next symbols to the M partitions randomly according to criterion given by Eq. (2), until all the partitions are mapped.

$$v(n_{ij}) = \left\lfloor \frac{p_j}{\sum_{j=1}^{K} p_j} \times M \right\rfloor + 1 \quad (2)$$

where $v(n_{ij})$ is the number of partitions mapped to n_{ij}.

C. Lossy Image Compression

For lossless joint compression and encryption, the reconstructed plaintext must be identical to the original one. No error or tolerance is allowed. However, for the lossy counterpart, certain amount of error or tolerance is allowed. This means that we don't need to find out the exactly matched symbol in the codebook. A nearby one is still acceptable. Under the framework of lossless joint compression and encryption mentioned above, this tolerance relaxes the restriction for the chaotic search trajectory to land on the partition exactly mapped to the symbol to be encoded. In fact, a single partition can be mapped to a group of symbols close to each other. If there is overlapping in the grouping of symbols, a particular symbol may belong to more than one group and the chance of getting the chaotic search orbit to land on the target partitions is higher. The corresponding number of iterations is fewer and the compression ratio is improved. In the decoding process, each partition maps to the mean value of the group of symbols, which is then used to represent all the symbols belonging to this group, similar to the principle of vector quantization.

There is another implementation of this relaxation. Instead of having multiple symbols mapped to a single partition, each partition is mapped to one symbol only. However, there is tolerance on the requirement of matching. For example, if the plaintext symbol is 97 and the chosen tolerance limit is 5, the chaotic trajectory landed on any one of the partitions mapped to symbols within the range of 92 to 102 is said to have hit the target. Again, the corresponding number of iterations is fewer and the compression ratio is raised.

IV. Simulation Results

To implement the proposed algorithm for joint operation of compression and encryption, the logistic map given by Eq. (3) is chosen as the chaotic map.

$$x_{n+1} = b x_n (1 - x_n) \quad (3)$$

The gain b is selected as 3.999999991 while the initial condition x_0 is 0.3388. The plaintext symbols are read in bytes and the chaotic map phase space is divided into 256 equal-width partitions. The maximum number of iterations for the search mode is chosen as 15. The proposed algorithm is implemented in C++ programming language running on a personal computer with an Intel Core2 2.13GHz processor and 1GB memory. The following data are collected.

To test the compression capability of the proposed scheme, the standard files from the Calgary Corpus are used [21]. There are 18 distinct files of different types, including text, executable, geophysical data and picture. Two simulation configurations are chosen for zero-order entropy. The first one is that only 16 more probable plaintext symbols are selected and they are all mapped to one table. In the second case, 128 more probable plaintext symbols are chosen and they are distributed to 16 tables, each has 8 symbols. The ciphertext-to-plaintext ratio (R) calculated by Eq. (4) is listed in Table 1.

$$R = \frac{\text{CiphertextLength}}{\text{PlaintextLength}} \times 100\% \quad (4)$$

TABLE 1: CIPHERTEXT-TO-PLAINTEXT RATIO OF THE CALGARY CORPUS FILES USING ZERO-ORDER ENTROPY.

File	1 map, 16 symbols	16 maps, 8 symbols each
pic	32.88%	31.43%
book1	83.36%	71.09%
paper2	83.95%	72.19%
paper3	84.30%	73.54%
paper4	84.27%	73.97%
progl	86.00%	74.82%
book2	85.05%	75.43%
paper5	86.30%	77.50%
progp	85.63%	77.73%
paper1	86.88%	78.38%
paper6	87.12%	78.81%
news	87.97%	81.73%
progc	88.74%	82.15%
bib	89.07%	82.34%
geo	83.98%	85.22%
trans	93.25%	86.98%
obj1	88.60%	89.62%
obj2	92.85%	94.03%

Table 1 shows that all files can be compressed using the two configurations. However, the compression performance of the second configuration (16 maps, 8 symbols each) is better for most of the files. The image file (pic) is the easiest to compress due to the redundancy of image pixels. The corresponding ciphertext length is about one-third the plaintext length. The executable files (obj1 and obj2) are the most difficult to compress as the distribution of plaintext symbols is comparatively uniform. The corresponding ciphertext sequences are only about 10% shorter than the plaintext ones.

The compression performance using first-order entropy can be found in Table 2. Eight next symbols, *i.e., K*=8, are selected. The data in the bracket of the rightmost column are the ratio when the necessary secret parameters are included as a header of the ciphertext. All the values are below 100% which imply that all the files can be compressed. The performance is the best for the pic file that the ciphertext length is only about one-third of the plaintext, due to the high redundancy in the picture.

TABLE 2: CIPHERTEXT-TO-PLAINTEXT RATIO OF THE CALGARY CORPUS FILES USING FIRST-ORDER ENTROPY.

File	Size (byte)	Ciphertext-to-plaintext ratio (With Header)
pic	513,216	31.67% (32.11%)
progp	49,379	68.31% (72.87%)
progl	71,646	69.49% (72.63%)
paper4	13,286	71.90% (88.76%)
paper5	11,954	72.08% (90.84%)
paper2	82,199	72.20% (74.94%)
trans	93,695	72.23% (74.64%)
obj1	21,504	72.71% (83.16%)
bib	111,261	72.81% (74.84%)
paper3	46,526	72.85% (77.68%)
progc	39,611	73.39% (79.08%)
paper6	38,105	73.49% (79.39%)
book1	768,771	73.50% (73.79%)
paper1	53,161	74.07% (78.30%)
book2	610,856	74.68% (75.05%)
obj2	246,814	74.74% (75.65%)
news	377,109	78.63% (79.23%)
geo	102,400	79.27% (81.47%)

The performance of our algorithm for lossy image compression is tested using eight standard 512 x 512 gray-scale images in bitmap format. The tolerance limit of the pixel value is set to 20. The ciphertext-to-plaintext ratio can be found in Table 3. It ranges from around 25% to 43% which shows that the compression performance is satisfactory. The corresponding peak signal-to-noise ratio (PSNR) of the reconstructed image is around 30dB.

TABLE 3: CIPHERTEXT-TO-PLAINTEXT RATIO OF LOSSY IMAGE COMPRESSION.

Image	Original Size (byte)	Ciphertext-to-plaintext ratio (With Header)
goldhill	262,144	24.80% (25.66%)
lena	262,144	25.65% (26.51%)
sailboat	262,144	27.18% (28.04%)
peppers	262,144	28.63% (29.49%)
frog512	262,144	31.48% (32.34%)
aerial	262,144	35.57% (36.43%)
barb	262,144	40.17% (41.03%)
baboon	262,144	41.91% (42.77%)

To evaluate the key sensitivity, encryptions using the all-mask mode for first-order entropy were performed by introducing a very small change in one of the secret parameters. Then the resultant ciphertext sequence is compared with the original one bit-by-bit and the percentage of bit change is calculated. For parameters b and x_0, the 15th digit after decimal point is changed from 0 to 1. For c_{-1}, the least significant bit (bit 0) is toggled from 0 to 1. The measured bit change percentages are 49.97%, 50.03% and 49.99% for b, x_0 and c_{-1}, respectively. All the data are close to 50% which indicate that the ciphertext is very sensitive to the key.

To measure the plaintext sensitivity, a bit is changed at different positions of the plaintext sequence while the key remains unchanged. The two resultant ciphertext sequences are compared bit-by-bit and the percentage of bit change is calculated. The results are 50.00% (bit change at the beginning of plaintext), 50.03% (middle) and 49.98% (end), respectively. They are all close to 50%, which imply that the ciphertext is very sensitive to the plaintext.

The randomness of the binary mask sequence is confirmed by the statistical test suite recommended by the U.S. National Institute of Standards and Technology (NIST) [22]. Ten sequences, each of 1,000,000 bits, are extracted for testing and they all pass the statistical tests including frequency, block-frequency, cumulative-sums, runs, longest-run, rank, and FFT.

V. CONCLUSION

An algorithm for the simultaneous compression and encryption using chaotic maps has been proposed. It can be applied to lossless data compression as well as lossy image compression. The effectiveness of the proposed scheme is confirmed by the satisfactory ciphertext-to-plaintext ratio using standard data and image files. Simulation results verify that the ciphertext is very sensitive to a tiny change in the key or the plaintext and so the security is maintained.

ACKNOWLEDGEMENT

The work described in this paper was fully supported by a grant from CityU [Project No. 7002070].

REFERENCES

[1] X. Liu, P. Farrell, and C. Boyd, "A unified code" *IMA - Crypto & Coding' 99, LNCS 1746*, pp. 84-93, 1999.

[2] M. Grangetto, E. Magli, and G. Olmo, "Multimedia selective encryption by means of randomized arithmetic coding," *IEEE Trans. Multimedia*, vol. 8, no. 5, pp. 905-917, 2006.

[3] J. Wen, H. Kim, and J. Villasenor, "Binary arithmetic coding with key-based interval splitting," *IEEE Signal Processing Letters*, vol. 13, no. 2, pp. 69-72, 2006.

[4] H. Kim, J. Wen, and J. Villasenor, "Secure arithmetic coding," *IEEE Trans. Signal Process*., vol. 55, no. 5, pp. 2263-2272, 2007.

[5] C. Wu and C. Kuo, "Design of integrated multimedia compression and encryption cystems," *IEEE Trans. Multimedia*, vol. 7, no. 5, pp. 828-839, 2005.

[6] J. Zhou, Z. Liang, Y. Chen, and O.C. Au, "Security analysis of multimedia encryption schemes based on multiple Huffman table," *IEEE Signal Processing Letters*, vol. 14, no. 3, pp. 201-204, 2007.

[7] R. Bose and S. Pathak. "A novel compression and encryption scheme using variable model arithmetic coding and coupled chaotic system," *IEEE Trans. Circuits and Sys. I,* vol. 53, no. 4, pp. 848-857, 2006.

[8] Bo Mi, X. Liao, and Y. Chen, "A novel chaotic encryption scheme based on arithmetic coding," to appear in *Chaos Solitons and Fractals*.

[9] S. Lian, J. Sun, J. Wang, and Z. Wang, "A chaotic stream cipher and the usage in video protection," to appear in *Chaos, Solitons and Fractals.*
[10] M.S. Baptista, "Cryptography with chaos," *Physics Letters A*, vol. 240, no. 1-2, pp. 50-54, 1998.
[11] W.K. Wong, L.P. Lee and K.W. Wong, "A modified chaotic cryptographic scheme," *Computer Physics Communications*, vol. 138, no. 3, pp. 234-236, 2001.
[12] K.W. Wong, "A fast chaotic cryptography scheme with dynamic look-up table," *Physics Letters A*, vol. 298, pp. 238-242, 2002.
[13] K.W. Wong, "A combined chaotic cryptographic and hashing scheme," *Physics Letters A*, vol. 307, no. 5-6, pp. 292-298, 2003.
[14] S. Li, G. Chen, K.W. Wong, X. Mou, and Y. Cai, "Baptista-type chaotic cryptosystems: problems and countermeasures," *Physics Letters A*, vol. 332, pp. 368 – 375, 2004.
[15] G. Alvarez, F. Montoya, M. Romera, G. Pastor, "Cryptanalysis of an ergodic chaotic cipher," *Physics Letters A*, vol. 311, pp. 172-179, 2003.
[16] G. Alvarez, F. Montoya, M. Romera, G. Pastor, "Cryptanalysis of dynamic look-up table based chaotic cryptosystems," *Physics Letters A*, vol. 326, no. 3–4, pp. 211-218, 2004.
[17] K.W. Wong, K.P. Man, S. Li, X. Liao, "A more secure chaotic cryptographic scheme based on dynamic look-up table," *Circuits, Systems and Signal Processing*, vol. 24, no. 5, pp. 571-584, 2005.
[18] J. Wei, X. Liao, K.W. Wong, T. Zhou, and Y. Deng, "Analysis and improvement for the performance of Baptista's cryptographic scheme," *Physics Letters A*, vol. 354, no. 1-2, pp. 101-109, 2006.
[19] D. Xiao, X. Liao, and K.W. Wong, "Improving the security of a dynamic look-up table based chaotic cryptosystem," *IEEE Trans. Circuits and Sys. II*, vol. 53, no. 6, pp. 502-506, 2006.
[20] K.W. Wong, S.W. Ho, C.K. Yung, "A chaotic cryptographic scheme for generating short ciphertext," *Physics Letters A*, vol. 310, no. 1, pp.67-73, 2003.
[21] ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus
[22] A. Rukhin, J. Soto, *et al,* "A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications," NIST Special Publication 800-22, http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html

Kwok-Wo Wong graduated with a BSc(EE) degree from The Chinese University of Hong Kong and a PhD degree from City University of Hong Kong. Currently, he is an Associate Professor in Department of Electronic Engineering, City University of Hong Kong. His research interests include chaos and cryptography.

Ching-Hung Yuen graduated with a Bachelor of Engineering in Computer Engineering from City University of Hong Kong in 2006. He is working as a research assistant in the Department of Electronic Engineering, City University of Hong Kong. His research interests include chaos and cryptography.