A More Secure Chaos Based Encryption System with High Compression Capability

Anjali Elizabeth Joseph,
M.Tech Student(CSE),
Viswajyothi College of Engineering and Technology, Muvattupuzha, Kerala, India.

AmelAustine,
Assistant Professor(CSE),
Viswajyothi College of Engineering and Technology, Muvattupuzha, Kerala,

ABSTRACT
The Baptista cryptosystem uses the logistic map to generate chaotic sequence. The encryption process of a given plaintext character consists of iterating the chaotic map until the state reaches the subinterval assigned to that symbol. The cipher text corresponding to the plaintext symbol is the number of iterations in the chaotic map. We propose a method to enhance the security and compression capability of Baptista-type cryptosystem. Unlike the present chaotic cryptosystem which uses the parameter and the initial condition of the chaotic map as the key, this cryptosystem uses an external key to generate the initial condition for the chaotic map. The lookup table used for encryption process is determined adaptively based on the frequency of occurrence of plaintext symbols and is dynamically updated in the searching process.

Keywords
Chaos, compression, cryptography, simultaneous compression and encryption.

1. INTRODUCTION
New age of communication networks especially the World Wide Web and mobile-phones networks, have vastly extended the limits and possibilities of communications and information transmissions. Related with these rapid developments, there is an increasing demand of the cryptographic techniques. The need of secure communication and the rapid increase of the size of multimedia files transmitted using network communications have determined the development of new techniques to protect both the information transmitted and bandwidth of the channel. This leads to an increasing interest in the search of combined operation of compression and encryption on the same set of data, i.e., carry out these two operations concurrently, instead of separately. There are two different research approaches in this area, to integrate compression in cryptographic schemes or to insert encryption into compression algorithms. It is usual to embed encryption in compression schemes such as arithmetic coding and Huffman coding since both cryptographic ciphers and entropy coders bear certain similarity in the sense of secrecy. An entropy coder is simple to be converted to a cipher by using a secret key to govern the statistical model used in coding. The decoder can follow the variations in the statistical model and construct the accurate output only if the secret key is on hand.

The process of embedding encryption into arithmetic coding mainly focused on the control of interval distribution using a secret key [4]–[7]. In case of Huffman coding, a method based on several Huffman tables was proposed [8]. However, these schemes are all compression oriented. Encryption is a supplementary feature but not the primary concern. Thus, the security level may not be acceptable. For example, the method based on multiple Huffman tables [8] only switch the left and right branches in the Huffman tree according to the key bits, without altering the tree structure. The code length remains unaffected. The arithmetic code with key-based interval splitting [3] as well suffers from known-plaintext attack [7]. Based on the security concern, it is worth to study the feasibility of combined operation of encryption and compression by incorporating compression in existing encryption schemes.

A chaos-based encryption scheme designed by Baptista [1] searches the plaintext symbol in the lookup table using a key-dependent logistic chaotic trajectory and treats the number of iterations on the logistic map as the cipher text. Though it bear from the problem of cipher text expansion that the cipher text length is typically about 1.5 to 2 times that of the plaintext length. An effort was made in [2] to include certain compression capability in the Baptista-type encryption scheme by adaptively constructing the lookup table according to the possibility of occurrence of the plaintext symbols. The cipher text is no longer than the plaintext, but the compression performance still far from the source entropy. This is because the chaotic search trajectory repeatedly lands on the interval corresponding to irrelevant source symbols, the number of iterations is bigger than necessary, and the compression ratio is not close to the source entropy. This cryptographic system is a symmetric key method using the essence of chaos. This chaotic cryptosystem uses the parameter and the initial condition of the chaotic map as the key to this cryptosystem. Thus the security purely depends on the two constant parameters and thus it is vulnerable to the bruteforce attack.

In this concise, dynamic lookup table is adopted in the joint compression and encryption scheme as proposed in [3], with the goal of improving the compression performance. The lookup table used for encryption is dynamically updated in the searching process. If a symbol reached during the visit by the chaotic trajectory is not the one being encrypted, it is removed from the current lookup table by assigning the partitions associated with this symbol to another non visited symbol. Thus the target symbol eventually associates with a large no of partitions, it can be searched in less no of iterations. The cipher text is reduced in size, and a better compression ratio is achieved. In this scheme, initial condition to the logistic equation is generating from a key generation procedure. This will in turn increase the security capability of this cryptographic system.

The rest of this paper is ordered as follows. In the next section, the Baptista-type chaotic cryptosystem is reviewed. The approach for embedding compression in this class of chaotic Cryptosystem with dynamic lookup table is described in Section III. Our approach to increase the security

capability by using a key generation scheme is described in Section IV Simulation results and analyses can be found in Section V. In the last section, conclusions will be drawn.

2. BAPTISTA-TYPE CHAOTIC CRYPTOSYSTEM

In recent years, there is an increasing tendency of designing ciphers based on chaos. The chaotic systems are based on chaotic equations and are highly sensitive to the initial condition and the system parameters. These properties are advantageous in security. In addition, the awareness on chaos and nonlinear dynamics can be useful in the field of cryptography. A typical chaos-based cryptographic method [1] was proposed by Baptista in 1998.

$$\mathbf{x_{n+1} = bx_n(1 - x_n)} \quad (1)$$

Simple 1-D chaotic maps like the logistic map and the tent map are normally used for data and document encryption. The phase space of the chaotic map is divided into a number of equal-width slots, each slot maps to a possible plaintext symbol uniquely. A secret chaotic trajectory produced from the key-dependent chaotic map parameters and initial condition is employed to search for the slot mapped to the plaintext symbol being encrypted. The duration 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 plain text mapping can be believed as a lookup table or a codebook for encryption. In the decryption side, the same secret chaotic search trajectory is re-established, and the correct plaintext symbols are recovered only if the same secret key and the lookup table are presented. Unlike the above chaotic encryption scheme which uses the parameter and the initial condition of the chaotic map as the key, this scheme uses an external key of variable length to generate the initial condition for the chaotic map. The encryption of each block of plaintext is dependent on the secret key. Thus in the next section, an algorithm for combining compression in Baptista-type chaotic cryptosystems is offered so that the ciphertext length is guaranteed shorter than or equal to that of the plaintext and a key scheme that provide high secrecy.

3.PROPOSED APPROACH

3.1 Basic Principle

In this proposed system, the dynamic lookup table is formed adaptively using the plaintext symbol frequency. First, the whole plaintext string is scanned once to find out the frequency of occurrence of each plaintext symbol. The phase space of the chaotic map is spitted into a number of equal-width slots, and the number of slots mapped to a particular symbol is proportional to its frequency of occurrence. The lookup table will be updated if the symbol is not found during the searching. If the slots just visited maps to a non target symbol, all the slots associated with that symbol need to be changed to another symbol with high probability. The advantage of having more slots for more probable symbols is that the chance for the chaotic trajectory to land on those slots is also higher. As a result, the required number of iterations is smaller and a small amount of bits are used to encode it. This leads to the compression capability.

The proposed chaotic encryption scheme is a symmetric key block cipher using the soul of chaos. Distinct from the basic Baptista type chaotic cryptosystem which uses the initial condition of the chaotic map as the key, this cryptosystem uses an external variable length key to generate the initial condition for the chaotic map. The encryption of each plaintext symbol is dependent on the secret key and, thus the encryption scheme becomes stronger against any reasonable attack that uses feedback techniques.

3.2 Key Generation

The proposed system uses an external variable length key to generate the initial condition for the chaotic map. The main property of chaotic map is sensitivity to the initial condition. Thus high secrecy can be achieved by utilizing this feature. The encryption of each plaintext symbol is dependent on the secret key and, thus the encryption scheme becomes stronger against any reasonable attack that uses feedback techniques. The secret key of this cryptographic system is an external key of size 128 bits. In order to generate the 128bit key from any variable length key values, here we use the MD5 algorithm which produces 128bit output for inputs of any length. The 128bit key for encryption/decryption process will be divided in blocks of 8-bitsK = K1K2 . . .K16as session keys. The values of the initial condition for the logistic chaotic map are produced on the basis of the 128bit secret key using the next formulas:

$$\mathbf{x_0} = \frac{\left((\mathbf{k1}) \oplus (\mathbf{k2}) \oplus (\mathbf{k3}) \oplus \ \ldots \oplus (\mathbf{k16})\right)}{\mathbf{256}} \quad (2)$$

This x_0 value will be used as the initial condition for the chaotic logistic map. Here K1, K2, K3…K16 are 8 bit blocks that used to produce the 128 bit key for the chaotic encryption process.

3.3 Encryption

3.3.1 Lookup Table Formation

Search the whole plaintext string once to find out the number of occurrence for each possible plaintext symbol. Then, sort the list in descending order and select the top M symbols, s1,s2,s3..sm. Partition the phase space of the chaotic map into N equal-width slots, with N>M. Starting from J=1 , map the selected plaintext symbols to the N slots according to the given equation, until all the partitions are mapped.

$$n(s_j) = \left\lfloor \frac{N * u(s_j)}{\sum_{i=1}^{m} u(s_i)} \right\rfloor + 1 \quad (3)$$

Where *u(sj)* and *n(sj)* are, respectively, the number of occurrence and the number of partitions mapped to symbol *sj*. The mapping should be arbitrary so that the slots mapped to a particular symbol spread over the whole phase space of the chaotic map. In general, the selection of the slots or mapping can be determined by iterating a chaotic map and trace the slots that the chaotic trajectory falls in.

3.3.2 Chaotic Encryption

Sequentially encrypt each plaintext symbol by searching in the lookup table using a secret chaotic trajectory generated with logistic map.Initially, check whether the

plaintext symbol to be encrypted is one of the *M* selected symbols. Then the logistic map will be iterated with the initial condition generated during the key generation. If the symbol being encrypted is found during iteration, the number of iterations of the chaotic map is considered as the cipher text. If not, the lookup table will be updated during the next iteration. If the slot fallen during iteration maps to non targeted symbol then all the slots in the phase space that point to that symbol will be replaced with then on visited symbol with the highest probability. on the other hand, it should be noticed that the slots can be arbitrarily assigned to other symbols to increase the difficulty of attack. In the next iteration, the chaotic trajectory keeps on to search the target symbol in the updated lookup table until the plain text symbol is found. Otherwise it will be encrypted using masking. The plaintext symbol is taken as output directly. In order to identity the use of masking for the current symbol, a special symbol chosen as zero number of iteration is employed. When the current plaintext symbol has been encrypted, the lookup table is again according to the symbols probabilities of occurrence. These operations are continued until all the symbols in the plaintext sequence have been processed. Thus the cipher text is formed from the chaotic encryption part.

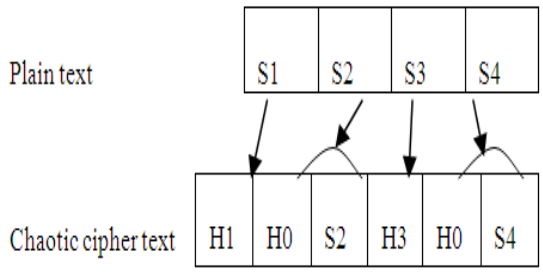

Fig 1: Chaotic Encryption

Fig 1 shows an example for chaotic encryption where S1 and S3 are encrypted in search mode using chaotic iteration and S2 and S4 in mask mode by appending header.

3.3.3 Compression

After all the plaintext symbols have been processed and the chaotic cipher text is produced, a Huffman tree is built for all the collected number of iterations, including zero. On the other hand, if the type of plaintext file is known, a characteristic Huffman tree for that file type can be used instead, and the time required for building the Huffman tree is saved. When the Huffman tree is built, the number of iterations and the special symbol are replaced by the corresponding variable-length Huffman code to form the intermediate sequence. If the size of the intermediate sequence is higher than that of the plain text size then plain text will be considered as the intermediate sequence.

3.3.4 Mask Encryption

The intermediate sequence should be masked by the binary mask sequence. Mask sequence is generated by iterating the chaotic map. During each iteration of the chaotic map, eight masking bits are extracted from the least significant byte of the chaotic map output *x* which is usually a double precision real number between0 and 1.The last eight bits of every *x* value is extracted and added to the mask *m*. In this process, both the binary mask sequence *m* and the intermediate sequence are divided into 32-bit blocks. The ciphertext of block is given by

$$C_i = \left(r_{i+1} \oplus m_{c_{i+1}+r_{i+1} mod(l/4)} \oplus C_{i-1}\right) mod\ 2^{32} \quad (4)$$

Where c_i and r_i are the *i*th 32-bit block in the ciphertext and intermediate sequences, respectively. The location of the mask block for $r_i$is determined by$c_{i-1 mod\ (L/4)}$,L where is the plaintext length in bytes. In order to encrypt the first block c_0, c_{-1} is needed, which is determined by the secret key.

3.4 Decryption

Before starting the decryption procedure, the key and the plaintext specific information must be sending to the receiver secretly. The secret key includes the parameters and the initial value of the chaotic map and also the initial cipher block. They are independent of the plaintext and can be chosen freely. On the converse, the plaintext-specific information includes the name and length of the plaintext file, the encryption mode, the probable plaintext symbols and their number of occurrence, and the Huffman tree. The table storing the mapping between the probable plaintext symbols and the phase space does not need to be transmitted because it can be reconstructed at the receiver from the key and the plaintext-specific information.

3.4.1 Mask Decryption

Iterate the chaotic logistic map using the shared secret parameters and the initial conditions to redevelop the chaotic trajectory that is used in encryption. Afterward, take out the mask bits from the binary representation of each chaotic map output to form the binary mask sequence. Unmask the ciphertext to get the output sequence. If all the plain text symbols are encrypted in mask mode, the output sequence is already the plaintext and no further steps are needed. Otherwise, it is the intermediate sequence that requires the next step for completing the decryption.

3.4.2 Decompression

The unmasked data is decompressed to get the chaotic encrypted cipher. By using the Huffman tree of the plain text that is shared between the sender and receiver, the decoding of the compressed data is performed. Since Huffman coding is a lossless compression scheme, exact output can be obtained.

3.4.3 Chaotic Decryption

The information of plaintext probability must be available to the receiver for recreating the plaintext symbol from the lookup table. If the number of iterations is smaller than the ciphertext it indicates that the chaotic search trajectory has not yet fall on the target symbol. Then, the symbol replacement process similar to the chaotic encryption scheme needs to be performed in order to coordinate with the encryption part. With the correct key, the original plaintext sequence can be reproduced from the lookup table. The decryption process is reverse of the encryption side. The decoder regenerates the chaotic trajectory from the secret key and then looks for the slot corresponding to the target plain text symbol. Similar to the encryption the decryption is also

performed sequentially .Thus the plaintext is securely transmitted from sender to receiver.

4. ANALYSIS

4.1 Compression Performance

Due to the visiting of non target symbols frequently in the original scheme, the number of iterations is large, and the compression performance is thus limited. In order to avoid the chaotic map trajectory falling into the slots corresponding to the unrelated symbols again, a dynamic table is employed in the modified scheme by replacing the visited symbol with a high probability symbol. Thus the number of iterations is reduced and compression performance is increased. In order to increase the compression capability lossless compression method Huffman coding is also used.

4.2 Security Analysis

The key of the proposed system is composed of the control parameter b and the initial value x_0 of the logistic map, together with the 32-bit initial cipher block. This system uses an external variable length key to generate the initial condition for the chaotic map. The main property of chaotic map is sensitivity to the initial condition. Thus high secrecy can be achieved by utilizing this feature. The encryption of each plaintext symbol is dependent on the secret key and, thus the encryption scheme becomes stronger against any reasonable attack that uses feedback techniques. The secret key of this cryptographic system is an external key of size 128 bits. Since this system uses variable length keys brute force attack will be very difficult and that will make the system more secure. Chaotic systems have the property of sensitivity to the initial conditions makes the system more robust against attacks.

5.CONCLUSION

The existing approach of incorporating compression with chaos-based encryption scheme suffers from low compression capability. In order to avoid this, proposed system uses a dynamic table which is random in nature and produce short cipher texts. The compression capability is progressed, which is close to that of conventional entropy coding. Unlike the present chaotic cryptosystem that makes use of the initial condition of the chaotic map as the key, this approach uses an external key to generate the initial condition for the chaotic map. Consequently the proposed chaotic encryption scheme produces higher compression ratio and secrecy.

6. REFERENCES

[1] M. S. Baptista, "Cryptography with chaos," *Phys. Lett. A*, vol. 240,no. 1/2, pp. 50–54, Mar. 1998.

[2] K. W. Wong and C. H. Yuen, "Embedding compression in chaos-based cryptography," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 11,pp. 1193–1197, Nov. 2008.

[3] Jianyong Chen, Junwei Zhou, and Kwok-Wo Wong," A Modified Chaos-Based Joint Compression and Encryption Scheme*" IEEE Trans. Circuits Syst. II, Exp. Briefs,, vol. 58, no.2, Feb 2011*

[4] X. Liu, P. Farrell, and C. Boyd, "'A unified code' IMA crypto & coding' 99," *Lect. Notes Comput.Sci.*,vol. 1746, pp. 84–93, 1999.

[5] 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, Oct. 2006.

[6] J.Wen, H. Kim, and J. Villasenor, "Binary arithmetic coding with keybased interval splitting," *IEEE Signal Process. Lett.*, vol. 13, no. 2, pp. 69–72, Feb. 2006.

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

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