CIRCUITS SYSTEMS SIGNAL PROCESSING
VOL. 24, NO. 5, 2005, PP. 571–584

DOI: 10.1007/s00034-005-2408-5

A MORE SECURE CHAOTIC CRYPTOGRAPHIC SCHEME BASED ON THE DYNAMIC LOOK-UP TABLE*

Kwok-Wo Wong,[1] *Kwan-Pok Man,*[1] *Shujun Li,*[2] *and Xiaofeng Liao*[3]

Abstract. We have proposed a chaotic cryptographic scheme based on iterating the logistic map and updating the look-up table dynamically. However, it has been broken recently. In this paper, the weaknesses of the original dynamic look-up table scheme are analyzed and a more secure chaotic encryption scheme based on this dynamic look-up table concept is proposed. Theoretical analysis indicates that the modified scheme can resist the reported attacks. Moreover, simulation results show that this encryption scheme leads to a flat ciphertext distribution.
Key words: Chaos, cryptography, logistic map.

1. Introduction

The growing research interest in chaos and cryptography has resulted in a number of digital chaotic cryptographic approaches that realize private key cryptography with chaos [1], [5], [12]. Among these chaotic cryptographic schemes, the one proposed by Baptista [5] has attracted much interest. Since its publication in 1998, a number of variants [6], [7], [9]–[11], [13]–[15] have been suggested. The concept of this type of chaotic cryptosystem is very simple. The interval of interest in the phase space of the chaotic map is divided into a number of partitions equal to the total number of possible plaintext blocks. Each plaintext block is encrypted as the number of iterations applied in the chaotic map in order

* Received February 15, 2005; revised May 30, 2005; The work described in this paper was fully supported by a grant from CityU [Project No. 7001555].
[1] Department of Computer Engineering and Information Technology, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon Tong, Hong Kong. E-mail for Wong: itkwwong@cityu.edu.hk; E-mail for Man: 50264150@student.cityu.edu.hk
[2] Department of Electronic Engineering, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon Tong, Hong Kong. E-mail: hooklee@hooklee.com
[3] Department of Computer Science and Engineering, Chongqing University, Chongqing 400044, People's Republic of China. E-mail: xfliao@cqu.edu.cn

to reach the partition correspondent to that block [5]. The resultant ciphertexts are integers and are suitable to be transmitted through public data communication networks.

There are two major drawbacks with Baptista's approach. First, the ciphertexts usually concentrate on a small number of iterations and so their distribution is not flat enough to ensure high security. Second, the cryptographic scheme is too slow to make it suitable for practical use, such as in the secure transmission of large multimedia files through the internet. Finally, the ciphertext size is usually long, at least twice that of the original plaintext. To deal with the first drawback, we have modified the original method so as to obtain a flatter ciphertext distribution [15]. To increase the encryption speed, we have proposed a fast approach using a smaller look-up table that is updated dynamically [13]. Moreover, a short ciphertext encryption scheme is suggested in [14] to reduce the length of the ciphertext.

A number of attempts for the cryptanalysis of the original Baptista-type chaotic cryptosystem and its variants have been performed [2]–[4], [8], [9]. The first attempt was reported in [8]. Only one attack was proposed and it is in partial form. In the analysis made in [2], it was found that this type of chaotic cryptosystem actually behaves as a stream cipher although it operates like a block cipher in that a fixed number of plaintext bits are encrypted to an integer number of iterations. Based on this observation, a keystream attack was applied to break the cryptosystem successfully. A similar attack [3] was used to break the variant for generating flat ciphertext distribution [15]. Our modification based on a dynamic look-up table [13] has also been broken recently [4]. The objective of this paper is to analyze the weaknesses of the original dynamic look-up table scheme and to propose a more secure chaotic encryption scheme based on this dynamic look-up table concept.

2. The original dynamic look-up table scheme

We first review the chaotic cryptographic method proposed in [13], which is based on a dynamic look-up table. As in Baptista's method [5], the chaotic map chosen is the simple one-dimensional logistic map governed by the following equation

$$x_{n+1} = bx_n(1 - x_n), \tag{1}$$

where b is the gain and $x_n \in [0, 1]$. An initial look-up table, containing the mapping of each possible plaintext block to equal-width partitions in the interval $[x_{\min}, x_{\max}]$ of the phase space of the logistic map, should be set in advance. This initial mapping can be in order or at an agreed setting. For the encryption of the ith input block, we let the logistic map iterate until the trajectory first falls into the region corresponding to the ASCII code of this block. Similar to Baptista's method, the iteration will continue if the current number of iterations is smaller than a predefined minimum number of iterations. This prevents cryptanalysis upon the loophole of zero or just a few iterations. On the other hand,

if the current number of iterations is large enough, it is sent immediately as the ciphertext and no random numbers need to be generated. This is an advantage to a resource-constraint computing environment such as smartcard because no additional hardware or software random number generators are required. Before encrypting the next message block, we have to update the look-up table dynamically by exchanging the ith entry e_i with another entry e_j. In [13], the interval v between these two entries is solely determined by the current value of x using the following formula

$$v = \left(\frac{x - x_{\min}}{x_{\max} - x_{\min}} \right) * N, \tag{2}$$

where N is the total number of entries in the table, $x_{\max}$ is the maximum value of x in the chosen phase space region, and $x_{\min}$ is the minimum value.

The interval obtained from (2) is used to locate the other entry e_j to be swapped with e_i as the value of j is equal to the sum of i and v. This sum may be larger than N. In this case, we have to perform a modulus operation so that the index increment is in a cyclic manner and the value of j is still within N. As a result,

$$j = i + v \mod N. \tag{3}$$

In order to make the look-up tables for consecutive plaintext blocks as unlike as possible, just swapping a single pair of entries in the table after the encryption of a single plaintext block is no longer enough. An approach for achieving this was proposed in [14]. Instead of just swapping one pair of entries in the table, we can swap more pairs by making use of the interval, v, between two swapping entries obtained from (2). Starting from the current entry i, we swap the entries at locations i and $(i + v \mod N)$. Then we continue to swap $(p-1)$ pairs of entries starting from the entry next to the one last visited, i.e., $(i + v + 1 \mod N) \leftrightarrow (i+2v+1 \mod N)$, $(i+2v+2 \mod N) \leftrightarrow (i+3v+2 \mod N)$, $\ldots$, $(i+(p-1)v + p - 1 \mod N) \leftrightarrow (i + pv + p - 1 \mod N)$, where p is the total number of pairs to be swapped during the encryption of each plaintext block. After that, the look-up table updating process is completed and we can start the iterations on the logistic map for the next block. Then we perform the p-pair swapping in the look-up table again by increment i and calculate a new value of interval. Notice that if the index i reaches the bottom of the table, it will start from the top again.

An illustration of this generalized dynamic updating of the look-up table is given in Figure 1. Suppose that the input block size is 8-bit. Therefore N equals 256 and there are 256 entries in the table, from e_0 to e_{255}. Moreover, $x_{\min}$ and $x_{\max}$ are chosen as 0.2 and 0.8, respectively. Suppose that we are now encrypting the fourth 8-bit input block. The current x value is 0.4395. By using (2) and (3), we obtain $v = 102$ and $j = 4 + 102 = 106$. The entry e_4 is swapped with e_{106}. If p is chosen as 3, we have to swap two more pairs, i.e., $e_{107} \leftrightarrow e_{209}$, $e_{210} \leftrightarrow e_{56}$, as shown in Figure 1.

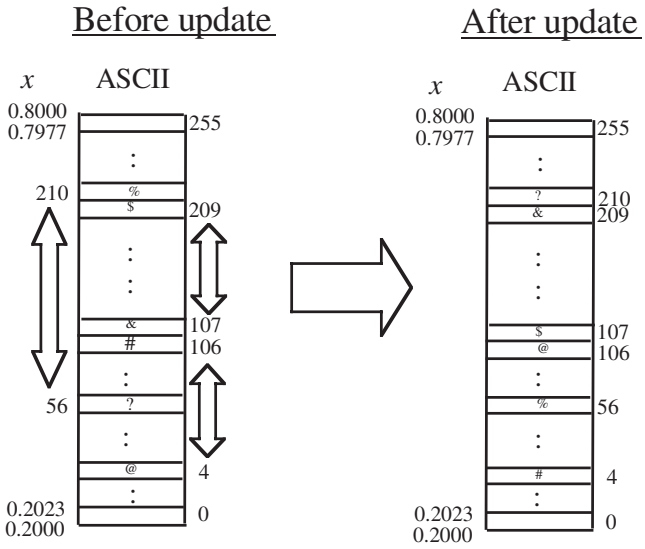

Figure 1. An illustration of the original dynamic updating of the look-up table as proposed in [13].

3. Weaknesses of the original dynamic look-up table scheme

There are a number of weaknesses in the dynamic look-up table scheme that make it vulnerable to cryptanalysis. As pointed out in [4], although the current chaotic state x is involved in (2) for calculating v, the output value actually depends solely on the plaintext, but not on the key. Therefore an attacker can recover the keystream used for encryption and so the cryptosystem is broken [4]. Second, the mapping of the plaintext block with the entries in the look-up table is fixed during the encryption of each plaintext block and the table update operation is performed only after the completion of the encryption of each plaintext block. Moreover, the swapping process starts from the first entry of the look-up table, which is fixed for all keys and plaintexts.

In order to enhance the security of the Baptista-type chaotic cryptosystem based on the dynamic look-up table scheme, the following modifications should be made:

1. The update of the look-up table must depend on the current chaotic state x.
2. The mapping of plaintext should be changed in the course of encrypting a plaintext block, but not after the completion of it.
3. The table update process should be done in the course of encrypting a plaintext block, but not after the completion of it.

4. The table update process should not start strictly from the first table entry, but should depend on the key.

In the next section we will propose a more secure dynamic look-up table chaotic encryption scheme that includes the above modifications.

4. The proposed scheme

The plaintext sequence is divided into a number of blocks of size M. The total number of partitions in the phase space of the chaotic map is twice the number of possible plaintext blocks, i.e., 2^{M+1}. If the plaintext sequence is read by bytes, the total number of partitions is 512. It is equal to 32 if the plaintext sequence is divided into a number of 4-bit blocks. The 2^{M+1} partitions are divided into two groups, half of them are marked while the other half are unmarked. This leads to an equal number of marked and unmarked partitions to achieve the highest entropy, i.e., most difficult to guess the location of the marked and unmarked partitions correctly. This look-up table is similar to the association map used in [6]. The 2^M marked partitions each associates uniquely with a possible plaintext block by a one-to-one mapping. The initial location of the marked partitions as well as the initial mapping is random and should be kept secret. A typical look-up table is shown in Figure 2(a) while the corresponding mapping with the plaintext block can be found in Figure 2(b).

The encryption procedures can start when the initial look-up table has been prepared. The first plaintext block is read and the chaotic map starts to iterate using the initial condition x_0. After each iteration, the value of the current chaotic state x will be checked. If it falls in an unmarked partition, this iteration is not considered as an effective iteration. Nothing will be modified and the iteration will continue until the chaotic orbit lands on a marked partition. Then the iteration is considered as an effective one and a counter of the number of effective iterations made in the encryption of this plaintext block will be incremented. If that marked partition corresponds to the current plaintext block, the number of effective iterations made in encrypting this plaintext block will be taken as the ciphertext.

On the other hand, if the marked partition that the chaotic orbit falls in does not correspond to the current plaintext block, some modifications on the look-up table should be done. However, these modifications should not be performed regularly in each landing so as not to favor cryptanalysis. They can be made intermittently such as updating the look-up table every three or five times of landing on the marked partitions. This updating can be realized by swapping the marking status of the current partition with the next partition with opposite marking status. This means that the current partition is changed from marked to unmarked. Then we search for an unmarked partition from the next partition and turn it as marked. In order to change the look-up table substantially, more pairs

Index	Marking Status
0	X
1	X
2	
3	
4	X
5	
6	
7	X
8	X
9	X
10	X
11	
12	
13	
14	X
15	

(a)

Plaintext Block	Index in Look-up Table
000	1
001	9
010	10
011	8
100	0
101	14
110	7
111	4

(b)

Figure 2. An initial look-up table with $M = 3$, i.e., each plaintext block contains three bits; The partitions with an "X" symbol are marked. (b) A mapping of the plaintext blocks with the index of the look-up table.

of swapping are required. From the partition that has just been marked, we search down for the next marked partition, make it unmarked and then mark the next unmarked partition. Notice that the search and swap operations are performed in a cyclic manner. This means that if the end of the look-up table is encountered, it will continue from the top.

After the mark and unmark processes, the total number of marked partitions in the table is still 2^M as we always swap pairs of marked and unmarked partitions. However, these 2^M newly marked partitions may no longer correspond to the original ones mapped to the 2^M plaintext blocks. Therefore we have to perform a remap operation, starting from the current partition that the chaotic orbit falls in. This means that the next marked partition is mapped to the all-zero plaintext block. Then we search in the look-up table and map the next marked partition to the plaintext block with all-zero but a "1" in least significant bit. This remap operation continues in a cyclic manner until all the newly marked partitions are associated with all the possible plaintext blocks uniquely. Note that the remapping process can be performed in a very quick manner: record all marked partitions in an array $U[0, \ldots, 2^M - 1]$ and use an index, i, to point to the partition corresponding to "0". In this way, the jth marked partition corresponds to the plaintext symbol $(j - i) \mod 2^M$.

The table update and remap processes are illustrated by an example. Assume that the table shown in Figure 2(a) is used and that the chaotic orbit falls into partition 4. Moreover, three pairs of marked and unmarked partitions have to be swapped. This means that we have to swap partitions 4 and 5 first. Then we go down and find that partition 7 is marked. This is swapped with partition 11. Finally, we swap partitions 14 and 15. The new look-up table is shown in Figrue 3(a). As the current chaotic orbit stays in partition 4, the next marked partition will be mapped to the all-zero plaintext block. Similarly, we map the remaining marked partitions with the corresponding plaintext block in order to obtain a new association table, as given in Figure 3(b).

Under this table update and remap scheme, the new look-up table depends on the current chaotic state x. This is because the start point of the swapping is the current partition that the chaotic orbit falls in. The latter is determined by x which is obtained by iterating the chaotic map, (1). The new look-up table will be different if the start point of the swapping is changed. This satisfies criterion (1) set in Section 3. The table update and remap processes are performed intermittently when the current partition that the chaotic orbit lands on is not the target partition. These conditions will be encountered a number of times before the chaotic orbit falls in the target partition finally, i.e., the completion of encrypting the current plaintext block. Therefore, criteria (2) and (3) stated in Section 3 are fulfilled. In the remap operation, the mapping starts from the current partition that the chaotic orbit lands on and so the last criterion set in Section 3 is also satisfied.

As the ciphertext is the number of effective iterations the chaotic orbit falls in before reaching the target partition, the probability of occurrence of small ciphertext is much higher than that of large ones. Therefore the ciphertext distribution will decrease with its magnitude and will not be flat. To achieve a flat ciphertext distribution, we can make use of the masking scheme proposed by Li et al. [9], [10]. However, here we do not choose to mask the current ciphertext using the information obtained from the encryption of the current plaintext block. This is

Index	Marking Status
0	X
1	X
2	
3	
4	
5	X
6	
7	
8	X
9	X
10	X
11	X
12	
13	
14	
15	X

(a)

Plaintext Block	Index in Look-up Table
000	5
001	8
010	9
011	10
100	11
101	15
110	0
111	1

(b)

Figure 3. (a) The resultant look-up table updated from the one shown in Figure 2(a); and (b) the corresponding new associate map. Assume that the chaotic orbit falls into partition 4 and three pairs of marked and unmarked partitions are swapped.

because there may be the chance that two or more combinations of the current plaintext and current chaotic state will result in exactly the same ciphertext, as analyzed in [10]. Instead we will use the information obtained from encrypting the previous plaintext block.

As described before, the updating of the look-up table will be performed inter-

mittently when the chaotic orbit falls in marked partitions. For the iterations that the chaotic orbit falls in marked partitions but the look-up table is not updated, we can perform a bitwise XOR operation of the index of the current partition accumulatively so as to store the partial history of the chaotic orbit. The resultant $(M+1)$-bit accumulated XOR value will be used to mask the next ciphertext. For the first ciphertext block, there is no such previous partial history and we can use a random $(M+1)$-bit sequence, i.e., an initial vector, instead.

The ciphertext is the number of effective iterations the chaotic orbit falls in before reaching the target partition. It could be a number much longer than an $(M+1)$-bit. In such a case, we have to divide the ciphertext into a number of $(M+1)$-bit blocks before masking with the $(M+1)$-bit last accumulative partial chaotic history. For example, if M is chosen as 4 but the current number of effective iterations is 62, there will be three 5-bit ciphertext blocks (31, 31, 0). Each block is masked with the 5-bit last accumulative partial chaotic history before sending out as the final ciphertext. The receiver should possess the exact knowledge of the last accumulative partial chaotic history before he/she can recover the correct number of effective iterations for decryption. As a result, this masking scheme not only leads to a flat ciphertext distribution, but also further enhances the security.

A block diagram of the whole encryption process is shown in Figure 4. The decryption process requires nearly the same operations as in encryption such as iterating the chaotic map, an update, and remap the look-up table intermittently. However, unmasking of the ciphertext blocks should be done at the beginning of each decryption so as to obtain the number of effective iterations required in finding the current plaintext block. Moreover, instead of checking whether the landed partition corresponds to the target one, the checking in the decryption process is whether the number of effective iterations has been reached. If this is the case, the decrypted plaintext is the one associated with the final partition the chaotic orbit lands on.

5. Simulation results

In order to test the effectiveness and the efficiency of the chaotic cryptographic scheme described in Section 4, it is proposed to encrypt and decrypt the following four types of source files:

File 1: audio (.mp3) file of size 98,304 bytes.
File 2: Word document (.doc) file of size 210,944 bytes.
File 3: executable (.exe) file of size 487,000 bytes.
File 4: video clip (.avi) file of size 1,087,430 bytes.

The encryption and decryption processes are implemented using the C++ programming language running on a personal computer with a Pentium IV 2 GHz processor and 512 MB RAM. The value of M is 4. This means that the plaintext

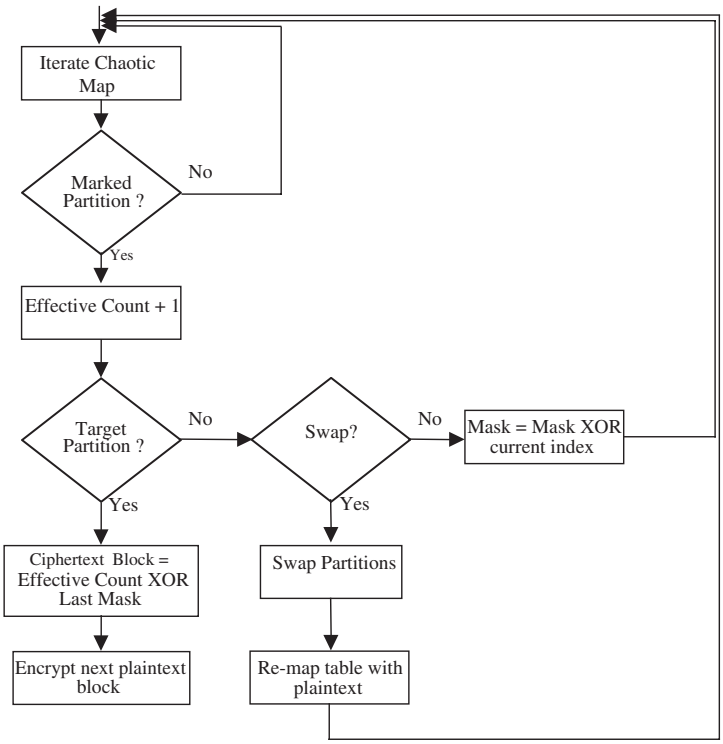

Figure 4. A block diagram of the whole encryption process.

file is read in blocks of length 4 bits. As a result, there is a total of 32 partitions in the look-up table and each ciphertext block is a 5-bit sequence. The value of b in (1) is selected as 3.9999995 while x_0 is chosen arbitrarily as 0.1777. The range of interest in the phase space is from 0 to 1, which is wider than the range from 0.2 to 0.8, as used in most Baptista-type cryptosystems. This choice increases the encryption speed but will not affect the distribution of ciphertext as the mapping is changed dynamically. The look-up table is updated every three times when the chaotic orbit falls in a marked partition. The initial mask is a 5-bit random sequence.

In Table 1, the encryption time, decryption time, number of iterations, length of ciphertext, and other statistics obtained from ten sets of randomly generated initial look-up tables are measured and listed in max–mean–min format. The total number of iterations means the number of times the chaotic map is iterated for one byte (8 bits) of plaintext. The number of effective iterations refers to the number of times the chaotic orbit falls into a marked partition during the encryption of one byte of plaintext.

Table 1. Performance of the proposed chaotic cryptographic algorithm*

	File 1 (.mp3) 98,304 bytes	**File 2 (.doc)** 210,944 bytes	**File 3 (.exe)** 487,000 bytes	**File 4 (.avi)** 1,087,430 bytes
Encryption time (s)	0.4–0.4–0.4	1.2–0.9–0.8	2.0–2.0–2.0	4.7–4.4–4.3
Decryption time (s)	0.4–0.4–0.4	1.1–0.9–0.8	2.0–2.0–1.9	4.3–4.3–4.2
Total number of iterations	539–74–2	661–70–2	682–74–2	681–72–2
Number of effective iterations	227–34–1	256–32–1	295–34–1	226–33–1
Ciphertext length (byte)	146,418–146,248–146,149	310,471–310,288–309,965	725,215–724,812–724,610	1,611,551–1,611,195–1,610,713
Ciphtertext to plaintext ratio	1.489–1.488–1.487	1.472–1.471–1.469	1.489–1.488–1.488	1.482–1.482–1.481

*All items are shown in max–mean–min format.

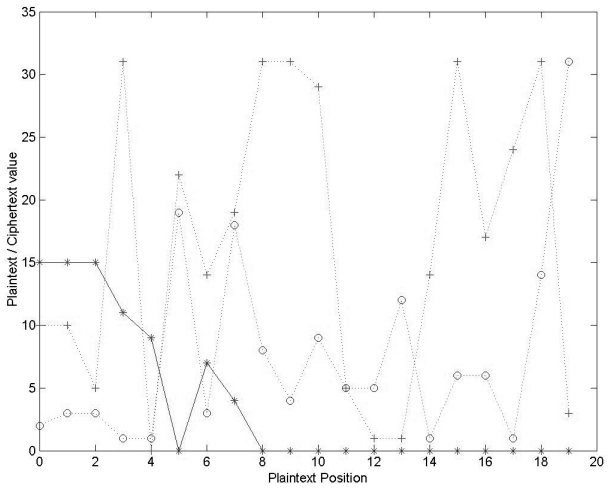

Figure 5. A plot of the plaintext and the two corresponding ciphertext sequences encrypted by two very close but not identical look-up tables (plaintext: *, ciphertext: + and ∘).

All the encrypted files are decrypted successfully. As nearly the same operations are performed at both the transmission and the receiving sides, the encryption and decryption processes require approximately the same amount of time. However, the decryption time is usually a little shorter. This is because looking-up in the association map is required in the encryption process, while the corresponding operation in the decryption process is just a check of whether the total number of effective iterations is equal to the desired one. The last row of Table 1 indicates that the ciphertext length is usually about 1.5 times that of the plaintext. This is shorter than the double-sized ciphertext obtained in most of the Baptista-type chaotic cryptosystems [5], [13], [15].

Theoretical analysis shows that the ciphertext depends substantially on the initial look-up table. This is verified here by plotting, in Figure 5, the values of the plaintext (the graph marked by $*$) and two corresponding ciphertext sequences (the graphs marked by $+$ and $\circ$), both with very close look-up tables except that a pair of marked and unmarked partitions are swapped. The graph shows that the resultant ciphertexts are totally different. Moreover, at positions 8 to 19, the plaintext value is zero but the corresponding ciphertext still appears randomly.

In order to analyze the distribution of the ciphertext, we have recorded the number of occurrences of each ciphertext block for the four input files using fixed b and x_0 but random initial look-up tables. The ciphertext distribution is plotted in Figure 6, which shows that the distribution is very flat due to the masking operation.

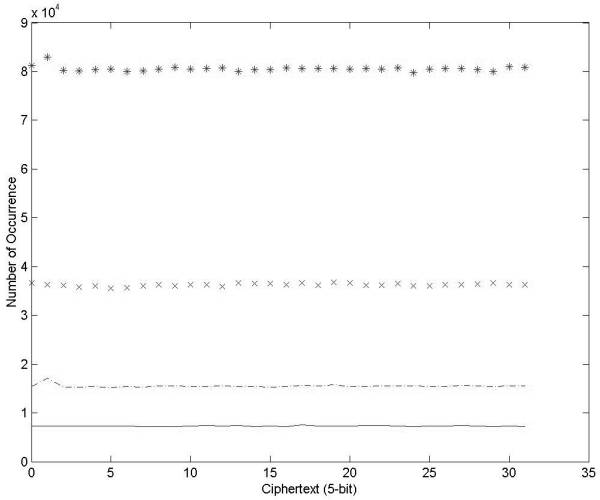

Figure 6. Distribution of the ciphertext. The graph from top to bottom corresponds to File 4, File 3, File 2 and File 1, respectively.

6. Conclusion

In summary, we have modified the original dynamic look-up table scheme [13], [14] so as to make the updating of look-up tables depend on the current chaotic state which in turn is determined by the key. It is believed that the modified scheme is more secure and can resist the attacks described in [4]. Moreover, simulation results show that this encryption scheme leads to a flat ciphertext distribution.

References

[1] E. Alvarez, A. Fernandez, P. Garcia, J. Jimenez, and A. Marcano, New approach to chaotic encryption, *Phys. Lett. A*, vol. 263, nos. 4–6, pp. 373–375, 1999.

[2] G. Alvarez, F. Montoya, M. Romera, and G. Pastor, Cryptanalysis of an ergodic chaotic cipher, *Phys. Lett. A*, vol. 311, nos. 2–3, pp. 172–179, 2003.

[3] G. Alvarez, F. Montoya, M. Romera, and G. Pastor, Keystream cryptanalysis of a chaotic cryptographic method, *Comput. Phys. Comm.*, vol. 156, no. 2, pp. 205–207, 2004.

[4] G. Alvarez, F. Montoya, M. Romera, and G. Pastor, Cryptanalysis of dynamic look-up table based chaotic cryptosystems, *Phys. Lett. A*, vol. 326, nos. 3–4, pp. 211–218, 2004.

[5] M. S. Baptista, Cryptography with chaos, *Phys. Lett. A*, vol. 240, nos. 1–2, pp. 50–54, 1998.

[6] F. Huang and Z.-H. Guan, Cryptosystem using chaotic keys, *Dhaos Soliton Fractals*, vol. 23, no. 3, pp. 851–855, 2005.

[7] F. Huang and Z.-H. Guan, A modified method of a class of recently presented cryptosystems, *Chaos Soliton Fractals*, vol. 23, no. 5, pp. 1893–1899, 2005.
[8] G. Jakimoski and L. Kocarev, Analysis of some recently proposed chaos-based encryption algorithms, *Phys. Lett. A*, vol. 291, no. 6, pp. 381–384, 2001.
[9] S. Li, X. Mou, Z. Ji, J. Zhang, and Y. Cai, Performance analysis of Jakimoski–Kocarev attack on a class of chaotic cryptosystems, *Phys. Lett. A*, vol. 307, no. 1, pp. 22–28, 2003.
[10] S. Li, G. Chen, K.-W. Wong, X. Mou, and Y. Cai, Baptista-type chaotic cryptosystems: Problems and countermeasures, *Phys. Lett. A*, vol. 332, nos. 5–6, pp. 368–375, 2004.
[11] A. Palacios and H. Juarez, Cryptography with cycling chaos, *Phys. Lett. A*, vol. 303, nos. 5–6, pp. 345–351, 2002.
[12] R. Schmitz, Use of dynamical systems in cryptography, *J. Franklin Inst.*, vol. 338, no. 4, pp. 429–441, 2001.
[13] K.-W. Wong, A fast chaotic cryptographic scheme with dynamic look-up table, *Phys. Lett. A*, vol. 298, no. 4, pp. 238–242, 2002.
[14] K.-W. Wong, S. S.-W. Ho, and S. C.-K. Yung, A chaotic cryptographic scheme for generating short ciphertext, *Phys. Lett. A*, vol. 310, no. 1, pp. 67–73, 2003.
[15] W.-K. Wong, L.-P. Lee, and K.-W. Wong, A modified chaotic cryptographic scheme, *Comput. Phys. Comm.*, vol. 138, no. 3, pp. 234–236, 2001.