Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 537934, 7 pages
http://dx.doi.org/10.1155/2013/537934

Research Article

A Parallel Encryption Algorithm Based on Piecewise Linear Chaotic Map

Xizhong Wang and Deyun Chen

School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China

Correspondence should be addressed to Xizhong Wang; ligongyan2011@sina.com

Received 22 April 2013; Revised 28 July 2013; Accepted 28 July 2013

Academic Editor: Xiaojie Su

We introduce a parallel chaos-based encryption algorithm for taking advantage of multicore processors. The chaotic cryptosystem is generated by the piecewise linear chaotic map (PWLCM). The parallel algorithm is designed with a master/slave communication model with the Message Passing Interface (MPI). The algorithm is suitable not only for multicore processors but also for the single-processor architecture. The experimental results show that the chaos-based cryptosystem possesses good statistical properties. The parallel algorithm provides much better performance than the serial ones and would be useful to apply in encryption/decryption file with large size or multimedia.

1. Introduction

There is no question that multicore processors have become the mainstream. Principal microprocessor manufacturers Advanced Micro Devices (AMD) and Intel released multicore chips for PCs, laptops, and servers. The main reason is the need for less heat and more energy efficiency. At first, the fastest chips were heating up faster than the average fan could cool them down. On the other hand, single-core chips rely on tightly packed power-hungry transistors to get the job done. A multicore chip is easier to cool because the CPUs are simpler and use fewer transistors. Thus, they use less power and dissipate less heat overall. Each multicore CPU can work on a different task at the same time. However, most of the multicore systems will require new tools, new algorithms, and a new way of looking at programming.

With the rapid development of the Internet, a lot of various digital documents, such as text, image, video, or audio, travel from one destination to another via the network line. Some of these documents might be sensitive and confidential therefore need to be protected. The most effective method is to encrypt the information so that the only authorized users with the key can decrypt them.

In recent years, a great deal of chaos-based cryptographic schemes has been proposed [1–5]. There exists an interesting relationship between chaos and cryptosystems. The chaotic properties can be found in the classic Shannon's paper on cryptography [6], for example, the ergodicity, the sensitivity to initial condition or control parameters, deterministic dynamics, and structure complexity [7–10]. Unfortunately, widely used traditional cryptographic algorithms, such as Data Encryption Standard (DES), Advanced Encryption Standard (AES) and some chaotic cryptosystems meet a single-core processor. Sequential algorithms assign the tasks to be run serially on the processor. How to take advantage of multicore processors is a challenge.

As the main goal of this paper, a parallel algorithm of chaotic cryptographic scheme using piecewise linear chaotic map and Fibonacci sequences is introduced. The PWLCM is the simplest kind of chaotic maps from the viewpoint of realization, which has many desired dynamical properties. The PWLCM has uniform invariant density and good correlation functions [11]. In fact, the PWLCM has been widely used in chaotic cryptosystems [12–21]. However, dynamical degradation destroys the uniform distribution of the key stream generated from the chaotic iterations of the PWLCM. When chaotic systems are realized in digital computers with finite computing precisions, most dynamical properties of chaos systems are different from the ones in the continuous field. The quantization errors are introduced into chaotic

evolution of digital chaotic systems at every discrete step. It makes pseudo-orbit depart from the real ones of the continuous field. Because of the sensitivity of chaotic systems to initial conditions and control parameters, the pseudo-orbit in finite precision can be distinguished from the theoretical ones even after a few numbers of iterations. Therefore, Li et al. pointed out that Zhou's chaotic cryptosystem was either not secure enough from strict cryptographic viewpoint [22]. The reason lies in the dynamical degradation of the computerized PWLCM. The dynamical degradation of digital chaotic systems reduces the security of the designed chaotic cryptosystem. In order to overcome this problem, random perturbation-based approach has been used in some applications [23, 24]. In this paper, we use the Fibonacci sequence to convert values of the piecewise linear chaotic map into secret keys.

A plaintext as a sequence is divided into blocks or fragments, which are encrypted/decrypted in multicore processors. In the parallel system, master process assigns tasks to slave processes, slave processes encrypted/decrypted blocks. Both the master and slave processes communicate data by using the Message Passing Interface (MPI).

As mentioned above, the traditional encryption algorithms are difficult to deal with large size of files and multimedia by using the computing sources of multicore processors. Besides, a new algorithm can be suitable not only for the traditional single processor but also for multicore processors, because some users could use the single processor for encryption, when the information might be decrypted at the multicore processors. Based on this observation, we propose a parallel chaotic cryptosystem. Our major contributions are highlighted below.

(1) The algorithm can fully exploit the computing sources of multicore processors, which is designed with master/slave communications with the MPI.

(2) The MPI can be realized in the parallel computers with distributed memory and shared memory. Therefore, the algorithm has the general-purpose characteristic for the parallel platform.

(3) To the best of our knowledge, for the first time, it is reported that the chaotic cryptosystem combines the piecewise linear chaotic map with the Message Passing Interface in the parallel model.

The rest of this paper is organized as follows. Section 2 gives the chaotic cryptographic scheme. In Section 3, we describe the overall framework of the proposed parallel encryption algorithm. Section 4 evaluates the performance of the parallel algorithm. The last section concludes the paper and gives some remarks.

2. Chaotic Cryptosystem

The chaos has an outer complex behavior produced by the internal random property of the nonlinear definite system, which is a pseudorandom movement while it looks like a random process. In particular, many pseudorandom number generators are based on chaotic maps. This approach produces a pseudorandom sequence from chaotic maps.

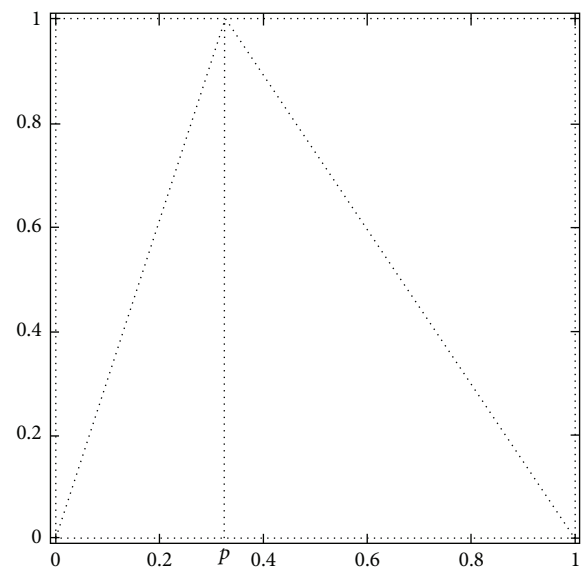


Figure 1: Skew tent map with a control parameter p.

A piecewise linear map is a map composing of multiple linear segments, where limited breaking points are allowed. A typical example of piecewise linear map is the skew tent map:

$$G(x) = \begin{cases} \dfrac{x}{p}, & x \in [0, p], \\ \dfrac{(1-x)}{(1-p)}, & x \in (p, 1], \end{cases} \tag{1}$$

where $p \in (0, 1)$ is the control parameter. For any control parameter $p \in (0, 1)$, the above piecewise linear map has a positive Lyapunov exponent and thus is always chaotic [7, 25].

Figure 1 shows the control parameter of the skew tent map. As long as the control parameter does not change the onto property of each linear segment, the obtained chaotic map will be good to use in chaotic cryptosystems.

Fibonacci sequence is employed to convert values of the skew tent map into integer number. Fibonacci sequence is denoted as the following equation:

$$f_{n+1} = (f_n + f_{n-1}) \mod Z; \quad n = 1, 2, \ldots, \tag{2}$$

where f_n and Z are a positive integer number.

Cryptosystems are typically divided into two generic types: symmetric key and asymmetric key. The symmetric key uses the same secret key both for encryption and decryption. The symmetric key is very fast and appropriate for handling large amounts of data. The proposed cryptosystem belongs to symmetric key, which is composed of the following two parts: the key generation and the encryption/decryption process. The simplified diagram of the cryptosystem is shown in Figure 2.

The secret key is described as follows:

$$k_n = (x_n \times f_n) \mod 256; \quad n = 1, 2, \ldots. \tag{3}$$

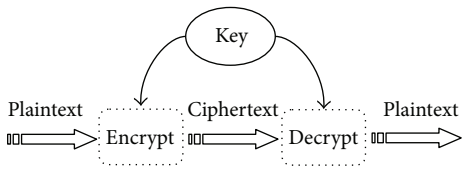


Figure 2: Diagram of the chaotic cryptosystem.

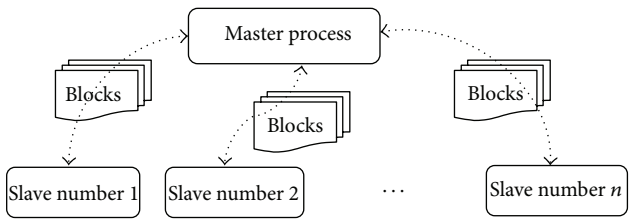


Figure 3: The client-server paradigm.

3. Parallel Algorithm

The parallel algorithm is implemented in a client-server paradigm. The tasks are allocated to a group of slave processes by a master process, which may also perform some of the tasks. Figure 3 shows that the client-server paradigm can be implemented in the multicore processors. A plaintext is divided into blocks or fragments, which will be encrypted or decrypted. As we know, a parallel client-server implementation might just run multiple copies of the code in slave processes after the master process assignes different blocks to each slave process. In other words, the master process will assign a new block to a slave process after the slave process finishes its task.

The parallel algorithm is designed with a master/slave communication model in which a designated master process controls, partitions, and distributes data to slave processes. We assume that the parallel system contains N_{bce} base core equivalents (BCE), where a single BCE represents a real CPU in multicore processors. Let N_{pvp} be the number of the parallel virtual processes (PVP) which include a master process and some slave processes. The parallel algorithm requires the master process to initialize parameters of the parallel system, for example, the control parameter of the chaotic map, initial values of the Fibonacci sequence and the chaotic map, the length of blocks, the number of PVP, and so on, before assigning works load to slave processes. Furthermore, the master process is responsible for input and output of the encryption/decryption file when reading the original file from a disk and writing the encrypted/decrypted file to a disk.

There exist three cases: $N_{\text{bce}} > N_{\text{pvp}}$, $N_{\text{bce}} = N_{\text{pvp}}$, and $N_{\text{bce}} < N_{\text{pvp}}$. At first, when $N_{\text{bce}} = N_{\text{pvp}}$, one of the BCEs runs the master process, which means *the master processor*. The other BCEs run slave processes and represent *slave processors*. In Figure 4, the parallel system has four parallel virtual processes which include the master processor and three slave processors. That is to say, each BCE acts as an individual processor. Communication between the master and slave processors uses the two MPI functions "MPI_Send" and "MPI_Recv," which are the basic point-to-point communication routines in MPI. For communication to occur, the sending processor must call MPI_Send and the receiving MPI_Recv, respectively. Note that, when $N_{\text{bce}} > N_{\text{pvp}}$, in the parallel system, some idle BCEs exist.

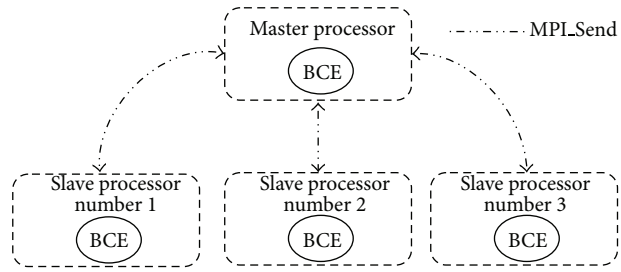


Figure 4: $N_{\text{bce}} = N_{\text{pvp}} = 4$.

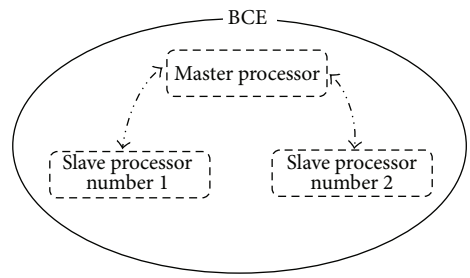


Figure 5: $N_{\text{bce}} = 1$, $N_{\text{pvp}} = 3$.

When $N_{\text{bce}} < N_{\text{pvp}}$, some BCEs might run several slave processes. In other words, one or more of slave processors are located in the same BCE so that the tasks are to be run in order.

In Figure 5, the single-core processor includes three PVPs, which are a master processor and two slave processors. The tasks, which should be implemented by the slave processors, run serially after the master processor assigned them. In particular, the proposed algorithm can be run not only on traditional machine or a single-core processor but also on the multicore processors.

The parallel algorithm is implemented by using the C programming language and the MPI library.

Algorithm 1. The encryption process in the slave process consists of the following steps.

Step 1. Receive the plaintext block from the master process (MPI_Recv).

Step 2. Generate $\{kn\}$.

Step 3. Encrypt the plaintext block.

Step 4. Copy the encrypted block to data buffer.

Step 5. Send the encrypted plaintext block to the master process (MPI_Send).

Algorithm 2. Master process consists of the following steps.

Step 1. Initialize MPI and parameters which include the initial value of x, the control parameter of p, Fibonacci sequence, the length of blocks, and so on.

Step 2_1. If the processor is master, send parameters of x, the control parameter, f_1, f_2, and the length of blocks, initial iteration number of the chaotic map and Fibonacci sequence, and so on to slave (MPI_Send).

Step 2_2. If the processor is slave, receive parameters from master (MPI_Recv).

Step 3. Initialize the chaotic map, Fibonacci sequence and create data buffer.

Step 4_1. If processor is master, master reads a block of original file from a disk.

Step 4_2. Send the block to slave processors (MPI_Send).

Step 4_3. Create data buffer in order to receive encrypted block from slave processors (MPI_Recv).

Step 4_4. Write the encrypted block to the disk.

Step 4_5. If the file pointer is the end of the file; end encryption process; else go to Step *4_1*.

Step 5. If processor is slave, execute the encryption process (Algorithm 1).

Step 6. System synchronization (MPI_Barrier).

Step 7. End MPI (MPI_Finalize).

Master process can determine the size, or number of the parallel virtual processes by using MPI function "MPI_Comm_size." A processor can determine its rank by using MPI function "MPI_Comm_rank."

4. Experimental Analysis

4.1. Chaotic Map. Generally speaking, there exists the dynamical degradation of digital chaotic systems. Therefore, the degradation lowers the security of the designed chaotic ciphers. In the C programming language, floating-point types include two sizes: float (single precision) and double (double precision).

Single-precision values with float type have 4 bytes. Double-precision values with double type have 8 bytes. In order to overcome the degradation of the computed finite precision, double type is used for computing values of the skew tent map. The long integer type is employed for calculating Fibonacci sequence. For determining the initial iteration number of the skew tent map, we simulate the chaotic sequences with slightly different initial values. Let N be iteration number.

Figure 6(a) shows the difference of the generated initial value x_0 between the case of x_0 and the case of $x_0 + 10^{-12}$ for $P = 0.527$. Figure 6(b) shows the difference with 10^{-12} for initial value x_0. The horizontal axis shows the number of iterations ($N = 100$), and the vertical axis shows the difference of the generated sequences. It is obvious that the initial iteration number at least is larger than 40.

The iteration number corresponds to the key sensitivity. The skew tent map has the sensitivity to changes in the control parameter p. Small variations of keys produce large changes by iterating the skew tent map. Therefore, cipher breaking becomes difficult by increasing the numbers of iterations. However, if the number of iterations is small, there is not much difference between the variations of the keys. It is undesirable to be used as keys.

We take 512×512 size 8 bits Lena's image as an example, where $P = 0.57$, $x_0 = 0.1$, $f_1 = 17$, $f_2 = 1$, and $Z = 60000$; the initial iteration number of the skew tent map and Fibonacci sequence is set to 100.

Original Lena's image and its histogram are shown in Figure 7. The encrypted Lena's image and its histogram are shown in Figure 8.

From Figure 8, we can see that the grayscale distribution of the encrypted image has a good balance property, which is secure against known plaintext attack.

4.2. Performance of the Parallel Algorithm. Amdahl's law is useful for analysing a system performance that results from two individual rates of execution, such as parallel or serial operations. Amdahl's law assumes that the computation problem size does not change when running on enhanced machines. In other words, the problem size remains the same when parallelized [26].

Amdahl's law states the following:

$$s = \frac{1}{(1-p) + (p/n)}, \tag{4}$$

where P is the proportion of a program that can be made parallel and $(1 - P)$ is the proportion that cannot be parallelized (remains serial); there exist n processors. The speedup of a program in multiple processors is limited by the time needed for the sequential fraction of the program.

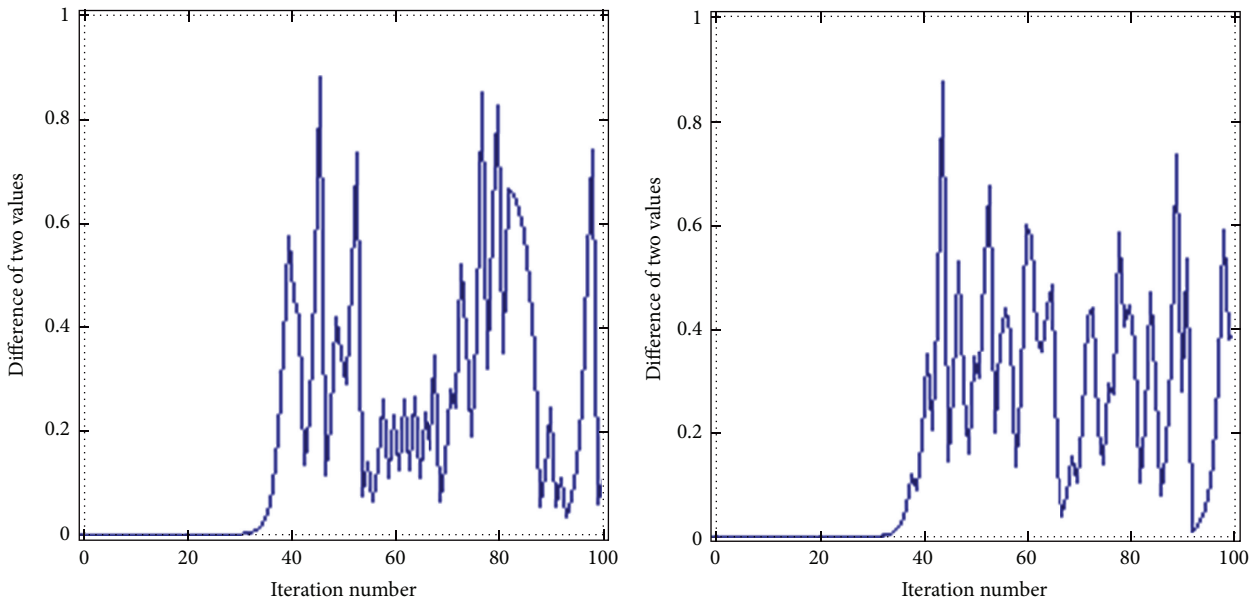
The time $T(n)$ an algorithm takes to finish when being executed on n thread(s) of execution corresponds to $T(n) = T(1)(E + (1 - E)/n)$, where n is the number of threads for execution and E is the fraction of the algorithm which is strictly serial.

The theoretical speedup that can be obtained by executing a given algorithm on a system capable of executing n threads of execution is $S(n) = T(1)/T(n)$.

In fact, the execution time T of a program in parallel system includes the initializing time T_{init}, the computing time T_{comp}, the communication time T_{comm}, and the system synchronization time T_{para}. It is described as follows:

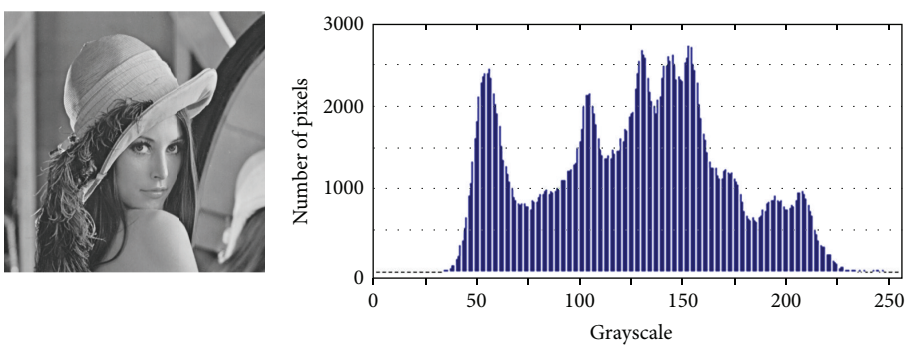
$$T = T_{\text{init}} + T_{\text{comp}} + T_{\text{comm}} + T_{\text{para}}. \tag{5}$$

T_{init} consists of the time of the initial iterating the chaotic map and the Fibonacci sequence, initializing data buffers,



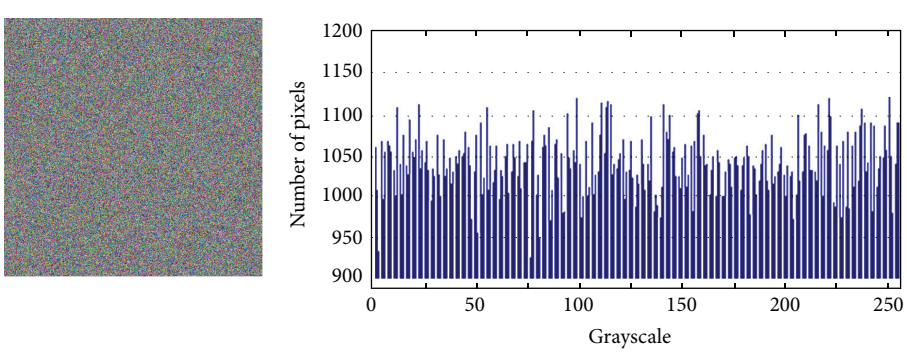
(a) (b)

Figure 6: Initial iteration number: (a) $x_0^1 = 0.1$, $x_0^2 = 0.100000000001$, $p^1 = p^2 = 0.527$; (b) $x_0^1 = x_0^2 = 0.1$, $p^1 = 0.527$, and $p^2 = 0.527000000001$.



(a) (b)

Figure 7: (a) Original image; (b) grayscale histogram.



(a) (b)

Figure 8: (a) Encrypted image; (b) its grayscale histogram.

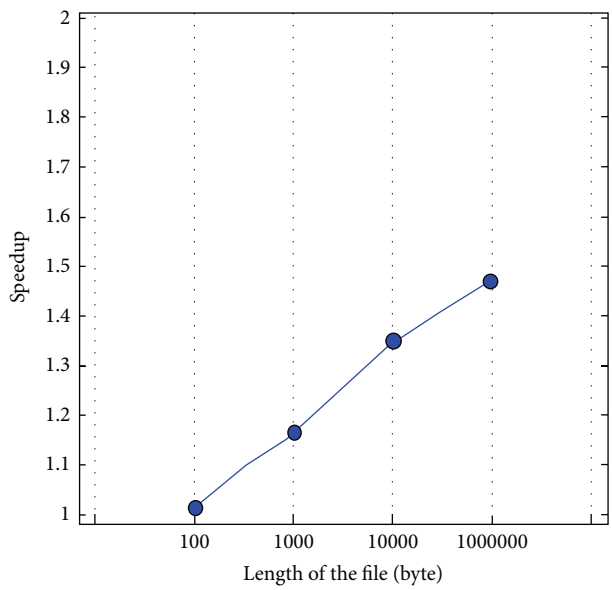


Figure 9: Speedup of the parallel algorithms.

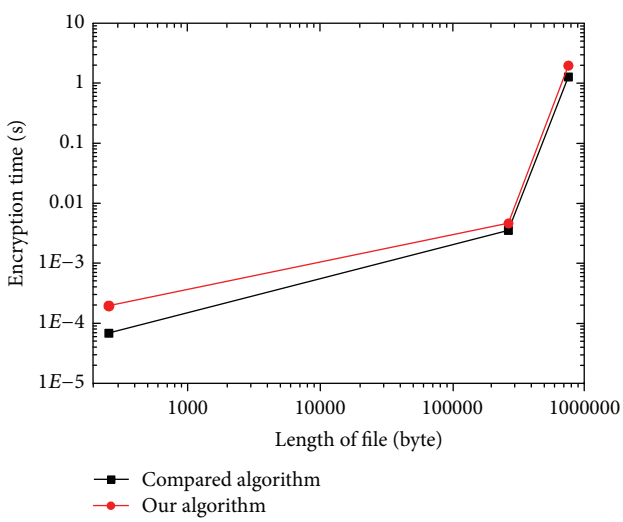


Figure 10: the proposed algorithm compared with existing methods.

and so on. We do not take account of the time of reading and writing files and blocks, because the different disks have different access times. The communication time and the system synchronization time depend on the parallel system, which might change when running on enhanced machines. According Amdahl's law, the execution time of a program is focused on the initializing time and the computing time.

The using memory of an algorithm is described as $M = M_{\text{lenFile}} + M_{\text{sysExe}}$, where M_{sysExe} is the memory for executing the algorithm, for example, to compute chaotic map and obtain keys. M_{lenFile} is the length of the file which is encrypted or decrypted. In the multicore processors, the memory architecture is the model of the shared memory. The length of the file for encryption (decryption) is the main part of the using memory. The greater the length of the encrypted file is, the larger the using memory will be.

In our experiments, the proposed algorithm was tested on a system equipped with Pentium Dual-Core CPU at 2.60 GHz, 2 GB RAM, and MS-Windows XP Professional. In Figure 9, the horizontal axis shows the length of a file, and the vertical axis shows the speedup obtained with lengths of the file: 100, 1000, 10000, and 1000000 bytes. For each execution time of a program, ten trial runs were conducted, and the total run time was obtained by computing the average. As it can be seen, the speedup of the algorithm slightly increases for the length of the file with 100 bytes. It can be explained by the fact that the computing time is a small part in the execution time of the program. In other words, the parallel proportion of a program is small. When the size of the file grows, the parallel proportion of a program increases.

The results show that, for larger length of a file, the parallel algorithm is more effective than the serial one.

In [27], the parallel algorithm is based on the logistic map. From Figure 10, we can see that our algorithm is better than the compared algorithm with a different length of files.

5. Conclusions

In this paper, we describe the parallelization of the chaos-based encryption algorithm. The algorithm is implemented in a client-server paradigm. It is suitable not only for multicore systems but also for the traditional single-core processor. The experiments show that the application of the parallel algorithm for multicore computers would considerably boost the time of the data encryption and decryption. We have confirmed that the parallel algorithm is of better performance than the sequential ones for encrypting or decrypting a larger size of the files, such as video, audio, and images.

Conflict of Interests

The authors declare that they have no conflict of interests.

Acknowledgments

The authors thank the anonymous referees and the editor for their helpful comments and suggestions. This research is supported by Heilongjiang Province Science Fund for Distinguished Young Scholars, China (Project approval no. JC201117).

References

[1] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps," *International Journal of Bifurcation and Chaos*, vol. 8, no. 6, pp. 1259–1284, 1998.

[2] M. S. Baptista, "Cryptography with chaos," *Physics Letters A*, vol. 240, no. 1-2, pp. 50–54, 1998.

[3] N. K. Pareek, V. Patidar, and K. K. Sud, "Discrete chaotic cryptography using external key," *Physics Letters A*, vol. 309, no. 1-2, pp. 75–82, 2003.

[4] Y. Mao, G. Chen, and S. Lian, "A novel fast image encryption scheme based on 3D chaotic baker maps," *International Journal of Bifurcation and Chaos*, vol. 14, no. 10, pp. 3613–3624, 2004.

[5] A. Akhshani, S. Behnia, A. Akhavan, H. Abu Hassan, and Z. Hassan, "A novel scheme for image encryption based on 2D piecewise chaotic maps," *Optics Communications*, vol. 283, no. 17, pp. 3259–3266, 2010.

[6] C. E. Shannon, "Communication theory of secrecy systems," *The Bell System Technical Journal*, vol. 28, pp. 656–715, 1949.

[7] G. Álvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *International Journal of Bifurcation and Chaos*, vol. 16, no. 8, pp. 2129–2151, 2006.

[8] T.-C. Lin and K.-W. Hsu, "A new adaptive H^{∞} control for chaos synchronization between two different chaotic systems," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 1, pp. 701–714, 2012.

[9] A. Alfi, "Particle swarm optimization algorithm with dynamic inertia weight for online parameter identification applied to lorenz chaotic system," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 2, pp. 1191–1203, 2012.

[10] H. Reza Koofigar and S. Amelian, "Adaptive nonlinear control of uncertain chaotic gyros subjected to unknown parameters and disturbances," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 9, pp. 6317–6327, 2012.

[11] A. Baranovsky and D. Daems, "Design of one-dimensional chaotic maps with prescribed statistical properties," *International Journal of Bifurcation and Chaos*, vol. 5, no. 6, pp. 1585–1598, 1995.

[12] H. Zhou and X. Ling, "Problems with the chaotic inverse system encryption approach," *IEEE Transactions on Circuits and Systems I*, vol. 44, no. 3, pp. 268–271, 1997.

[13] H. Zhou, X. Ling, and J. Yu, "Secure communication via one-dimensional chaotic inverse systems," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '97)*, vol. 2, pp. 1029–1032, June 1997.

[14] T. Sang, R. Wang, and Y. Yan, "Perturbance-based algorithm to expand cycle length of chaotic key stream," *Electronics Letters*, vol. 34, no. 9, pp. 873–874, 1998.

[15] T. Habutsu, Y. Nishio, I. Sasase, and S. Mori, "A secret key cryptosystem using a chaotic map," *IEICE Transactions E*, vol. 73, pp. 1041–1044, 1990.

[16] E. Alvarez, A. Fernández, P. Garcia, J. Jiménez, and A. Marcano, "New approach to chaotic encryption," *Physics Letters A*, vol. 263, no. 4–6, pp. 373–375, 1999.

[17] M. Jessa, "Data encryption algorithms using one-dimensional chaotic maps," in *Proceedings of IEEE Internaitonal Symposium on Circuits and Systems*, vol. 1, pp. 711–714, May 2000.

[18] S. Papadimitriou, T. Bountis, S. Mavroudi, and A. Bezerianos, "A probabilistic symmetric encryption scheme for very fast secure communication based on chaotic systems of difference equations," *International Journal of Bifurcation and Chaos*, vol. 11, no. 12, pp. 3107–3115, 2001.

[19] X. Yi, C. H. Tan, and C. K. Siew, "A new block cipher based on chaotic tent maps," *IEEE Transactions on Circuits and Systems I*, vol. 49, no. 12, pp. 1826–1829, 2002.

[20] P. García and J. Jiménez, "Communication through chaotic map systems," *Physics Letters A*, vol. 298, no. 1, pp. 35–40, 2002.

[21] N. Masuda and K. Aihara, "Cryptosystems with discretized chaotic maps," *IEEE Transactions on Circuits and Systems I*, vol. 49, no. 1, pp. 28–40, 2002.

[22] S. Li, X. Mou, Y. Cai, Z. Ji, and J. Zhang, "On the security of a chaotic encryption scheme: problems with computerized chaos in finite computing precision," *Computer Physics Communications*, vol. 153, no. 1, pp. 52–58, 2003.

[23] H. Zhou and X. Ling, "Realizing finite precision chaotic systems via perturbation of m-sequences," *Acta Electronica Sinica*, vol. 25, no. 7, pp. 95–97, 1997.

[24] T. Sang, R. Wang, and Y. Yan, "Clock-controlled chaotic keystream generators," *Electronics Letters*, vol. 34, no. 20, pp. 1932–1934, 1998.

[25] S. Li, G. Chen, and X. Mou, "On the dynamical degradation of digital piecewise linear chaotic maps," *International Journal of Bifurcation and Chaos*, vol. 15, no. 10, pp. 3119–3151, 2005.

[26] G. Amdahl, "Validity of the single processor approach to achieving large-scale computing capabilities," in *Proceedings of the Spring Joint Computer Conference (AFIPS '67)*, vol. 30, pp. 483–485, 1967.

[27] J. Liu, D. Song, and Y. Xu, "A parallel encryption algorithm for dual-core processor based on chaotic map," in *Proceedings of the 4th International Conference on Machine Vision: Computer Vision and Image Analysis; Pattern Recognition and Basic Technologies (ICMV '11)*, vol. 8350 of *Proceedings of SPIE*, 83500B, January 2011.