Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2014, Article ID 178585, 11 pages
http://dx.doi.org/10.1155/2014/178585

Research Article

An Effective Privacy-Preserving Algorithm Based on Logistic Map and Rubik's Cube Transformation

Wenbin Yao,[1,2] Pengdi Ye,[1,2] and Xiaoyong Li[1,2]

[1] *Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing 100876, China*
[2] *School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China*

Correspondence should be addressed to Wenbin Yao; yaowenbin_cdc@163.com

Received 24 April 2014; Revised 4 November 2014; Accepted 10 November 2014; Published 27 November 2014

Academic Editor: Daniele Fournier-Prunaret

Security and privacy issues present a strong barrier for users to adapt to cloud storage systems. In this paper, a new algorithm for data splitting called EPPA is presented to strengthen the confidentiality of data by two-phase process. In EPPA, data object is organized to be several Rubik's cubes executed for several rounds transformation at the first phase. In every round, chaotic logistic maps generate pseudorandom sequences to cover the plaintext by executing Exclusive-OR operation to form the cipher. Then logistic map is used to create rotation policies to scramble data information based on Rubik's cube transformation. At the second phase, all cubes are unfolded and combined together as a cross-shaped cube, which will be partitioned into a few data fragments to guarantee that every fragment does not contain continuous bytes. These fragments are stored on randomly chosen servers within cloud environment. Analyses and experiments show that this approach is efficient and useable for the confidentiality of user data in cloud storage system.

1. Introduction

As cloud services become more and more popular, securing data information within cloud environments has become a critical requirement [1]. The importance of minimizing information leakage in a cloud environment is highlighted by the current use of the cloud infrastructure for applications that require strong confidentiality guarantees.

Cloud environment denotes an architectural shift toward thin clients and conveniently centralized provision of computing resources. However, users' lack of direct resource control in the cloud prompts concern about the potential for data privacy violations, particularly abuse or leakage of sensitive information by service providers. The confidentiality in cloud systems is a big obstacle for users to step into it, as many users said "my sensitive corporate data will never be in the Cloud" [2].

In traditional researches, cryptographic algorithms are employed to ensure data confidentiality, such as OceanStore [3], FarSite [4], and PAST [5]. However, there is a tendency among users to keep passwords simple and memorable [6–8] leading to the possibility of brute force attacks, and keys that provide adequate encryption today are likely to be cracked by computation effort in the future.

Rabin designed information dispersal algorithm (IDA) [9] to deal with the security and reliability problem for the storage and transmission of data file in distributed systems. Nevertheless, it may stop malicious users from parsing information from one piece, and it still cannot prevent them from getting information from pieces. Therefore, Rabin's algorithm is always combined with keyed encryption by some distributed storage system.

The research of Subbiah and Blough [10] avoids the use of keyed encryption by using secret sharing threshold scheme that was introduced independently by Shamir [11] and Blakley [12]. Secret sharing is mostly used in distributed storage system, but if the attackers have the opportunity to get enough parts, the original object still can be reconstructed.

Potshards [13, 14] separates security and redundancy by utilizing two levels of secret sharing, while Subbiah and Blough's work only uses one level of secret sharing. It uses secret sharing to split an object into fragments at the first level.

Then this scheme uses redundancy encoding to make shares. However, the algorithm has a storage overhead problem produced by two-level secret sharing.

Jakimoski and Kocarev [15–17] discussed the relationship between cryptography and chaos theory and similarities of their crucial concepts such as mixing property and sensitivity to changes in initial conditions and parameters. They also present a systematic procedure for design of encryption algorithms based on chaotic maps.

Parakh and Kak [18–20] use an approach that creates data partitions and distribute them over the network within cloud environment, which they called implicit security. Thus, data objects might be fragmented in such a way that one partition reveals no usable information to the attackers. The authors admit this method may be inefficient for large datasets, so they provide an alternate data security method which encrypts the data and stores it on a single server that data owner trusts. Then the partitions of the key are created and dispersed over the network to keep encryption key secret. However, when the attackers get these partitions, the key will be reconstructed and lead to information leak directly. However, it suffers from the drawback that the staff member generating the keys can access the entire data set and must be trusted not to reveal the keys to others.

ESSA [21] is presented as a two-level information dispersal scheme, which takes advantage of knight's tour problem as scramble policy. Because of figuring out tour paths may need a lot of computation, this approach needs to obtain and store all of tour paths before scrambling data. As the path number of knight's tour on a 8×8 matrix is almost 3.3×10^{13} [22], if recording one tour path needs 128 bytes, it will cost about 4 PB storage space to save all of these tour paths, which is unaffordable for most information systems.

Gentry [23] discovers a fully homomorphic encryption scheme based on ideal lattices. However, it remains far from being adopted in practice due to the expansion in cipher text and slow encryption and decryption speeds. A number of fully homomorphic encryption schemes have been discovered since then [24–27] however still suffer from the practical limitations.

Kaddoum et al. [28, 29] focus on giving a good performance to chaos-based communication system, while other researchers [30–33] leverage Rubik's cube and logistic sequence to provide image encryption. However, these researches are not for the cloud storage environment.

The goal of this paper is not to provide a general encryption algorithm or privacy-preserving computation algorithm such as homomorphic encryption but rather offer a data privacy-preserving algorithm called EPPA, which combines data encryption and partition to address the problem of data confidentiality in cloud storage environment.

Data object in EPPA is encrypted and partitioned into fragments and stored at randomly chosen servers in the cloud that are known only to the owner of the data. Access to data objects depends not only on the knowledge of initial value of logistic map but also on the knowledge of pieces combination order and where these pieces are stored. It scrambles data information in two phases to strengthen the confidentiality of data.

In the first phase, original data objects are split into segments. Every segment is organized as an $n \times n \times n$ scramble cube. Every scramble cube carries out r round rotations. Each round executes two operations, which are data encryption and scramble. Two mapping rules, which are XOR table mapping and rotation method table mapping, are used to map the sequences that are created by logistic map to XOR tables and rotation method tables.

On the basis of XOR tables, the shifted subsquares are XORed to implement encryption. Then according to the rotation method tables, the data information covered on the subsquares of the scramble cubes may be scrambled several times based on the principles of Rubik's cubes rotations. In this phase, the original data is split and rearranged, and thus the former semantics becomes meaningless. Though data information is scrambled, it still has locality feature. Therefore, the locality feature will be broken in the next phase.

In the second phase, all of the scrambled Rubik's cubes are unfolded and combined together as a cross-shaped cube, which will be split into fragments to guarantee that every fragment does not contain continuous bytes so that none of the individual fragment will reveal any useful information. At last, these fragments will be stored, respectively, on randomly chosen servers of cloud computing system. In this phase, data object is split and recombined to break the data locality feature, so that malicious users cannot parse any semantics information even if they obtain some fragments.

Analyses prove the confidentiality of EPPA and the experiments show that EPPA has decent run-time efficiency.

The rest of this paper is organized as follows. Section 2 presents system model and security model. Section 3 introduces logistic map, Rubik's cube, and the rotation model of Rubik's cube. Section 4 describes the algorithm details of EPPA. Section 5 conducts a series of analyses and experiments to the EPPA's confidentiality and run-time efficiency. The last section concludes our work.

2. System Model and Security Model

2.1. System Model. Figure 1 shows the architecture of EPPA, which consists of the following system entities:

(1) a cloud provider which manages and operates a cloud infrastructure of storage services,

(2) cloud users that employ the cloud storage resource facilities to remotely store data. The Internet is the main communication backbone for exchanging information between the cloud users and the cloud.

In this system model, the key which is kept by the cloud user consists of the fragments order, fragments location information, and the initial value of logistic map. EPPA processes data objects in two phases guaranteeing that every fragment does not contain any continuous bytes. Then it chooses several servers randomly and sends one fragment to each chosen server. At last, it records the fragments' location

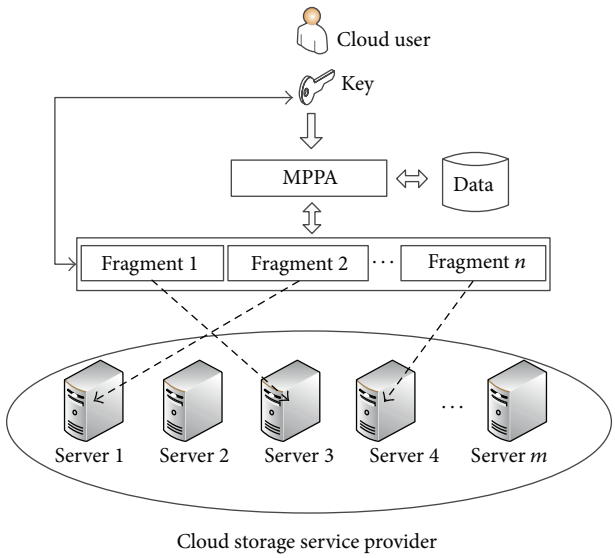

Figure 1: System overview.

information and combination order, which are needed when users try to read data.

2.2. Security Assumption. EPPA has the following assumptions.

(i) The cloud users' environment is trusted and will not be corrupted by any adversary. Users keep fragments order, fragments location information, and the initial value of logistic map themselves.

(ii) The cloud service provider is semitrusted. It will not deny service to any authorized users and will correctly execute the tasks assigned by users. Besides, it has access control policy, so that users cannot access unauthorized data. However, it is curious about the data content.

2.3. Security Model. The security model of EPPA is described by summarizing the possible attacks and delimiting the antiattack capability.

Brute Force Attack. EPPA should provide large key space so that the attackers cannot try all possible combinations in acceptable time to guess the key used to decrypt fragments stored in the cloud.

Statistical Attack. A small change in the key of EPPA should cause a drastic change to resist statistical attack.

Differential Cryptanalysis. EPPA should provide extremely low differential approximate probability so that the attackers cannot discover where the cipher exhibits nonrandom behavior, exploiting such properties to recover the secret key.

Linear Cryptanalysis. EPPA should provide extremely low linear approximate probability so that the attackers cannot find affine approximations to the action of a cipher.

3. Logistic Map and Rubik's Cube Transformation

A discrete dynamical system is a dynamical system whose state evolves over state space in discrete time steps according to a fixed rule, which can be described as a mapping G of a phase space x to itself. Therefore, the logistic map and Rubik's cube, which are employed in EPPA, are all discrete dynamical systems.

3.1. Logistic Map. Chaos and cryptography have some common features, the most prominent being sensitivity to variables' and parameters' changes. Iteration of encryption roundleads to the desired diffusion and confusion. Iteration of the chaotic map spreads the initial region over the entire phase space.

An important difference between the encryption round and the chaotic map is that the encryption round is defined on a finite set and depends on the key value. This is not the case within the chaotic map. A mapping defined on a finite set can be derived from a chaotic map by discretization, in which EPPA substitutes the continuous variables and the operations defined over real numbers with variables that take values from the finite set of integers and appropriate integer operations.

Researches [34–36] numerically observe that the average cycle and transient lengths of a chaos derived pseudorandom number generator (PRNG) grow exponentially with the precision of implementation and from this fact deduce that using high-precision arithmetic one can obtain PRNGs which are still of cryptographic interest.

Logistic map is one of the most widely used chaotic maps, which has distinctive properties for encryption, such as the natural concealment and high sensibility to initial condition. It is employed to generate rotation policies for Rubik's cubes and XOR table for encryption, the function of which is shown as

$$x_{n+1} = \mu x_n \left(1 - x_n\right). \tag{1}$$

By using formula (1) iteratively, a sequence can be created. In order to keep the iterate sequence chaotic from almost all of initial conditions, $\mu = 4$ is in common use. According to the initial value sensibility of logistic map, no same sequences will be generated while two initial values are different. Consequently, the initial value could be regarded as a secret key for encryption and decryption.

3.2. Rubik's Cube. Rubik's cube is a kind of toy, which is divided into several subcubes. In $n \times n \times n$ Rubik's cube, one sticker covers one subsquare, so $n \times n$ stickers cover each of the six squares. Every column can be rotated clockwise or anticlockwise around one axis. By turning the columns, a certain picture can be pieced together on the surface of the Rubik's cube.

3.3. Rubik's Cube Rotation Model. One row or column rotation of Rubik's cube can be represented as the movement of

several subcubes, which can be decomposed into two parts, subcube's position shift and subsquare's orientation shift.

3.3.1. Subcube's Position Shift. The process of subcube's position shift can be defined as

$$C' = RC. \tag{2}$$

Parameters C and C' indicate the state of subcube before and after rotation, respectively. R indicates state transition matrix of one rotation of Rubik's cube.

A Rubik's cube can build three-dimensional coordinates like Figure 1 shows, and the position coordinate of one subcube can be described as follows:

$$C = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{3}$$

From (2) and (3), the process of subcube's position shift can be represented by

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{4}$$

Figure 2 uses $3 \times 3 \times 3$ Rubik's cube as an example to describe the three-dimensional coordinates of every subcube. The central point of Rubik's cube is regarded as the origin of coordinates. Then coordinate of every subcube on the surface of Rubik's cube can be represented by $[x \ \ y \ \ z]$, $x, y, z \in \{1, 0, -1\}$.

According to three-dimensional coordinates in Figure 2, substitute any subcube's position coordinate before rotation and position coordinate after rotation into (4), and the state transition matrix of clockwise rotation around x-axis can be figured up by

$$R_x = C'C^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{5}$$

Anticlockwise rotation around x-axis can be figured up by the same way:

$$R_{x\text{-anti}} = C'C^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{6}$$

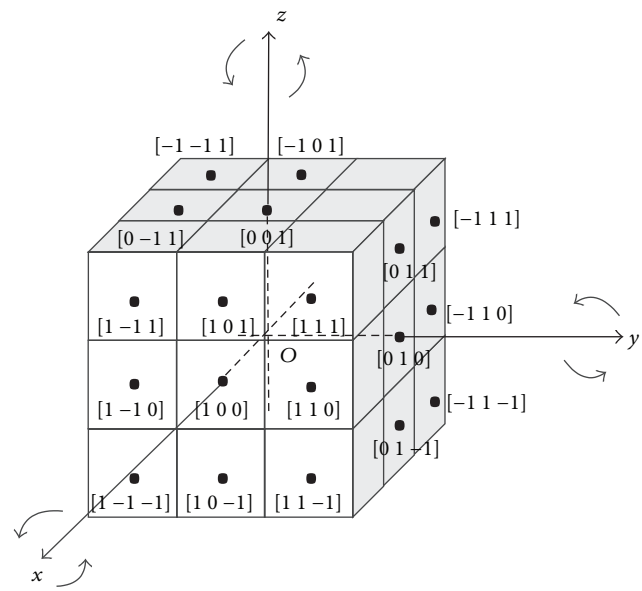

Figure 2: $3 \times 3 \times 3$ Rubik's cube.

R_y, $R_{y\text{-anti}}$, R_z, and $R_{z\text{-anti}}$, which can be figured out by the same way, are described as follow:

$$\begin{aligned} R_y &= \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \\ R_{y\text{-anti}} &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \\ R_z &= \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\ R_{z\text{-anti}} &= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{7}$$

3.3.2. Subsquare's Direction Shift. As Figure 3 shows, six different direction squares of Rubik's cube are marked as Up (U), Down (D), Front (F), Back (B), Left (L), and Right (R).

While a row or column rotates around one axis, its subsquares that are parallel to this axis will change their direction. Figure 3 shows an example that if a column rotates around y-axis, then the subsquares in red color should change their direction. Define a function:

$$D_x: U \longrightarrow L \longrightarrow D \longrightarrow R \longrightarrow U \tag{8}$$

which means that when a column makes clockwise rotation around x-axis, the subsquare in Up direction will change its direction to Left, the subsquare in Up direction will change its direction to Left, the subsquare in Left direction will change its direction to Down, the subsquare in Down direction will change its direction to Right, and the subsquare in Right direction will change its direction to Up.

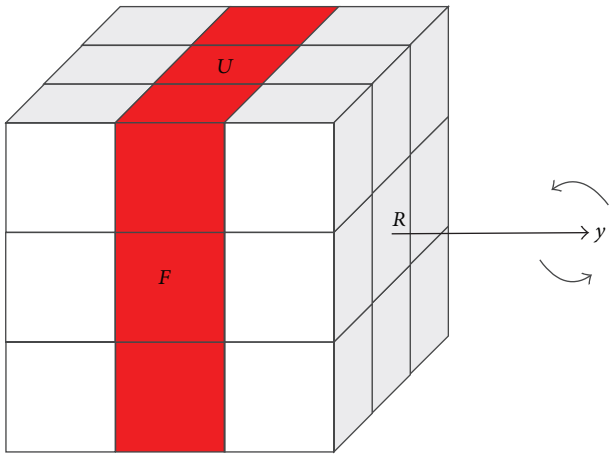

Figure 3: $3 \times 3 \times 3$ scramble cube with 54 bytes of data.

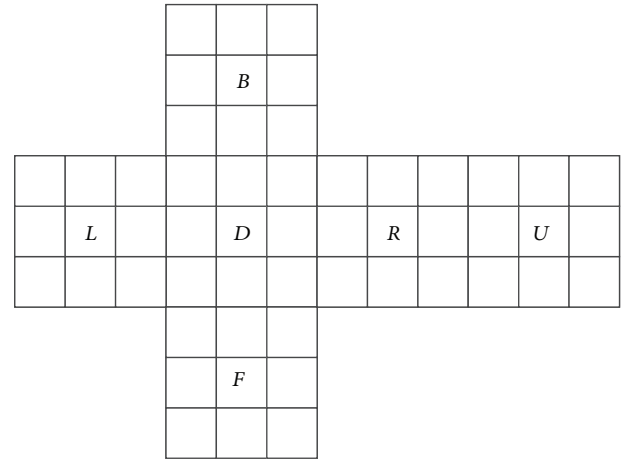

Figure 4: Unfolded $3 \times 3 \times 3$ scramble cube.

Other functions can be defined in the same way:

$$\begin{aligned} D_{x\text{-anti}}&: U \longrightarrow R \longrightarrow D \longrightarrow L \longrightarrow U \\ D_{y}&: U \longrightarrow B \longrightarrow D \longrightarrow F \longrightarrow U \\ D_{y\text{-anti}}&: U \longrightarrow F \longrightarrow D \longrightarrow B \longrightarrow U \\ D_{z}&: F \longrightarrow L \longrightarrow B \longrightarrow R \longrightarrow F \\ D_{z\text{-anti}}&: F \longrightarrow R \longrightarrow B \longrightarrow L \longrightarrow F. \end{aligned} \tag{9}$$

4. Details of EPPA

4.1. Definitions

Definition 1 (segment). EPPA treats the original data object as byte stream and splits it into fixed length pieces, which are recorded as segment. Every byte of a segment is attached to the subsquare of Rubik's cube. Therefore, each segment has $6 \times n \times n$ bytes.

Definition 2 (scramble cube). Scramble cube is an $n \times n \times n$ Rubik's cube that contains data from a segment on its surfaces. Data information on the cube will be scrambled by Rubik's cube rotation. Figure 3 shows a $3 \times 3 \times 3$ scramble cube with 54 bytes.

Definition 3 (split cube). Every three-dimensional cube can be unfolded by opening the cube from the Up direction square. Figure 4 shows the flat pattern of an opened cube. Arranging all of the unfolded scramble cubes one by one as a cross-shaped cube, as shown in Figure 5, forms a split cube. Figure 6 indicates a split cube combined by three unfolded scramble cubes, and each unfolded scramble cube is parallel to *YOZ* coordinate plane.

Definition 4 (fragment). Fragment is the data piece made by split cube. The subcubes with red color in Figure 6 composed a typical fragment. EPPA divides original data objects into $6 \times n \times n$ fragments.

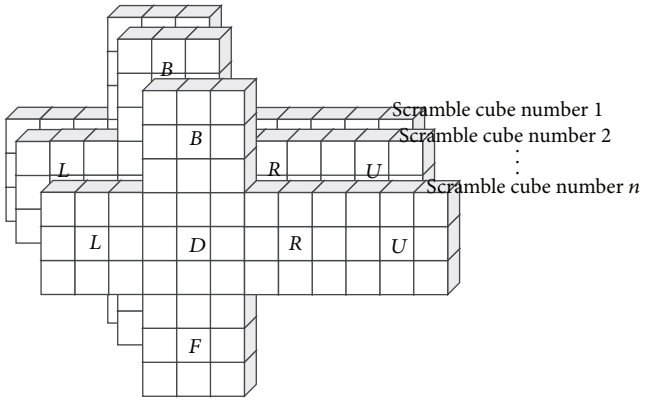

Figure 5: Unfolded scramble cube arrange method.

4.2. Algorithm Details. Figure 7 shows the details of EPPA, which is mainly combined by two-phase process.

In the first phase, original data objects are split into segments. Then this segment is organized as an $n \times n \times n$ scramble cube.

Two mapping rules, which are XOR table mapping and rotation method table mapping, are used to map the sequences that are created by logistic map according to XOR table and rotation method tables.

Every scramble cube carries out r round rotations. Each round executes two operations, which are data encryption and scramble. According to XOR tables, the shifted subsquares are XORed to implement data encryption. Besides, on the basis of the rotation method tables, the data information covered on the subsquares of the scramble cubes may be scrambled several times based on the principles of Rubik's cubes rotations. In this phase, the original data is encrypted and scrambled, and thus the former semantics becomes meaningless. However, data information still has locality feature. Therefore, the locality feature will be broken in the next phase.

In the second phase, all of the scrambled Rubik's cubes are unfolded and combined together as a cross-shaped cube,

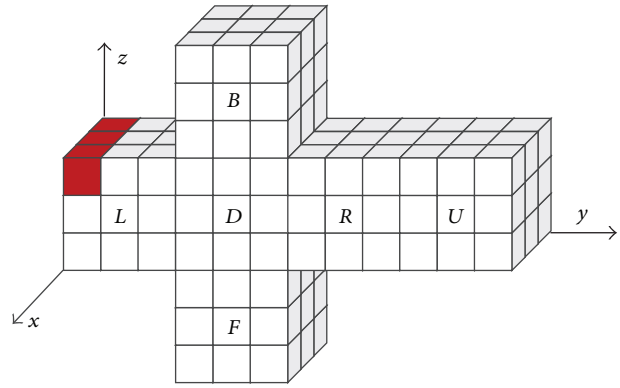

Figure 6: A split cube with 3 unfolded scramble cubes.

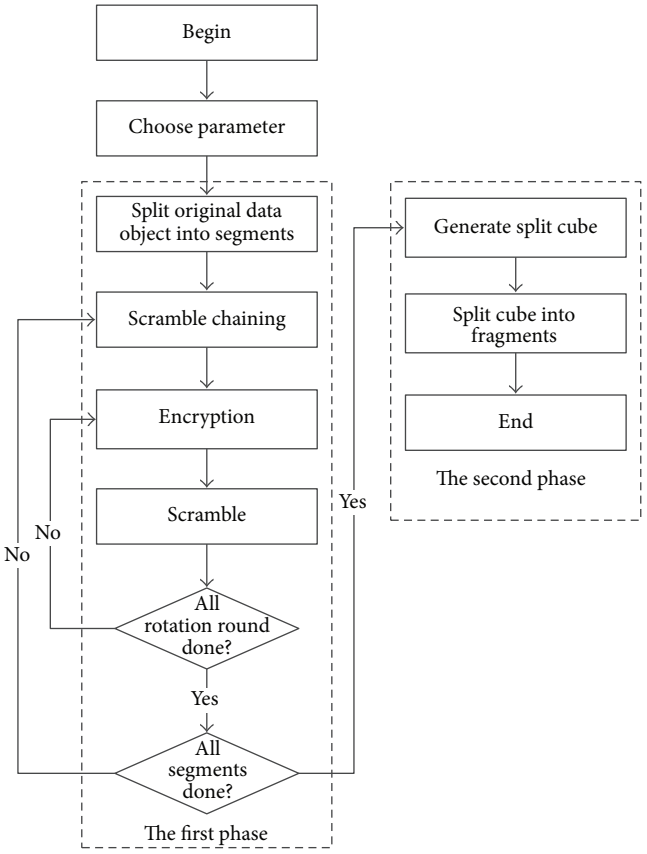

Figure 7: EPPA progress.

which will be split into fragments to guarantee that every fragment does not contain continuous bytes. At last, these fragments will be stored, respectively, on randomly chosen servers of cloud storage system. In this phase, data object is split and recombined to break the data locality feature, so that malicious users cannot parse any semantics information even if they only obtain some fragments.

4.2.1. Initialization. EPPA needs users to choose the split parameters. Then the system can figure out the scramble method. The split parameters that need to be determined are as follows:

(i) the size of Rubik's cube is $n \times n \times n$, $n \geq 3$;

(ii) the initial value of logistic map is $x_0^0 \in (0, 1)$;

(iii) the size of rotation round is r.

Users can choose n based on the need of fragments' number, which is $6 \times n \times n$.

4.2.2. First Phase. The first phase is composed by segment chaining and r round transformation. Every round transformation has two operations, which are data encryption and data scramble based on the rotation of scramble cube.

(1) Segment Chaining. In this step, when segment S_i finishes its rotation round, it will be XORed with segment S_{i+1}. After that, segment S_{i+1} can start its own rotation round. A sequence with $6 \times n \times n$ element is created by using formula (1) iteratively with initial value of logistic map x_0^0, whose element is uniformly partitioned and mapped to integers within $\{0, \ldots, 255\}$. It can be treated as segment S_0 to complete segment chaining with segment S_1.

(2) Mapping Rules. EPPA leverages logistic map to create XOR table and rotation method table for every scramble cube. Formula (10) is defined to generate a sequence by using formula (1) iteratively:

$$X_i = \left\{x_i^1, x_i^2, \ldots, x_i^r\right\}. \tag{10}$$

Parameter $i \in \{1, \ldots, N/(6 \times n \times n)\}$ that represents this sequence is prepared for the ith scramble cube, which is created from the last element x_{i-1}^r in X_{i-1} by formula (1). Parameter r indicates the size of rotation round.

Since the output of logistic map X_n belongs to $[0, 1]$, discretization is essential process to map X_n to Rubik's cube rotation policy and replace schedule. As a result, the logistic map is scaled so that input and output values of the map are in the interval. Meanwhile, the scaled logistic map is discretized. Two mapping functions of rotation method vector mapping and replacing table mapping rules are needed to achieve target.

(a) XOR Table Mapping. Since EPPA uses XOR operation to encrypt data information, a XOR element set can be indicated as $\{0, 1, 2, \ldots, 255\}$. Because every segment has $6 \times n \times n$ bytes data, a sequence with $6 \times n \times n$ elements is needed and will be mapped to the element within $\{0, \ldots, 255\}$ to complete XOR operation.

When in the jth round rotation of Rubik's cube, x_i^j is used as initial value to generate r different logistic sequence $X_{ij} = \{x_{ij}^1, x_{ij}^2, \ldots, x_{ij}^{6\times n\times n}\}$, $j = 1, 2, \ldots, r$, by formula (1). Then the XOR table can be created by the mapping function below:

$$W_{ij} = M_W\left(X_{ij}\right) = \text{floor}\left(4x_{ij}\left(1 - x_{ij}\right) \times 256\right). \tag{11}$$

Function M_W maps every element in sequence X_{ij} to an integer that in the range $\{0, \ldots, 255\}$. Then the XOR table $W_{ij} = \{w_1, w_2, \ldots, w_{6\times n\times n}\}$ is constructed.

(b) Rotation Method Table Mapping. As $n \times n \times n$ Rubik's cube has three axes, and every axis has two rotation directions, which are clockwise and anticlockwise, there will be $2 \times 3 \times n$ basic rotation methods. A rotation tuple (m_a, R, D) is defined to represent a rotation method, in which parameter $m \in \{1, \ldots, n\}$ means the mth row or column perpendicular to a ($a \in \{x, y, z\}$) axis, parameter R means the state transition matrix, and parameter D means the direction shift function.

So a rotation method set with $6 \times n$ element can be described as $\{(1_x, R_x, D_x), (1_x, R_{x\text{-anti}}, D_{x\text{-anti}}), (2_x, R_x, D_x), \ldots, (n_x, R_{x\text{-anti}}, D_{x\text{-anti}}), (1_y, R_y, D_y), (1_x, R_{y\text{-anti}}, D_{y\text{-anti}}), \ldots, (n_y, R_{y\text{-anti}}, D_{y\text{-anti}}), (1_z, R_z, D_z), (1_z, R_{z\text{-anti}}, D_{z\text{-anti}}), \ldots, (n_z, R_{z\text{-anti}}, D_{z\text{-anti}})\}$.

Therefore, a random sequence, whose elements are integral numbers within $[1, 6 \times n]$, is needed in order to map numbers to the $6 \times n$ different rotation methods.

Rotation method vector mapping function for each scramble cube is created as follows:

$$V_i = M_V(X_i), \tag{12}$$

where $i \in \{1, \ldots, N/(6\times n\times n)\}$, N means the number of bytes of original data object. M_V maps every element in X_i to an integer that in the range $[1, 6 \times n]$ and then maps the integers to the rotation method set. In this way, the rotation method table $V_i = \{v_1, v_2, \ldots, v_r\}$ is constructed. Every element in V_i is a rotation tuple (m_a, R, D). Since every segment is treated as a scramble cube, which may be rotated several times based on the rotation methods table, the scramble cube can be rotated several times. As a result, the segments are scrambled.

(3) Round Transformation. After generating scramble cube from segment S_i, rotation method table $V_i = \{v_1, v_2, \ldots, v_r\}$ can be created by rotation method table mapping, and XOR table $W_{ij} = \{w_1, w_2, \ldots, w_{6\times n\times n}\}$, $j = 1, 2, \ldots, r$, can be created by XOR table mapping. According to v_j and W_{ij}, $j = 1, 2, \ldots, r$, data encryption and scramble can be described as follows.

(a) Data Encryption. Let $B_{j,k}$, $k = 1, 2, \ldots, 6 \times n \times n$ denote every byte in one segment. On the basis of XOR table $W_{ij} = \{w_1, w_2, \ldots, w_{6\times n\times n}\}$, the partial encryption is given with

$$\begin{aligned} B_{j,2} &= B_{j-1,1} \oplus f_0 \\ B_{j,3} &= B_{j-1,2} \oplus f_1 \\ &\vdots \\ B_{j,0} &= B_{j-1,6\times n\times n} \oplus f_{6\times n\times n-1} \\ B_{j,1} &= B_{j-1,0} \oplus f_{6\times n\times n}, \end{aligned} \tag{13}$$

where the functions f_k have the following form:

$$f_k = B_{j-1,1} \oplus \cdots \oplus B_{j-1,k} \oplus w_{j,k}. \tag{14}$$

(b) Data Scramble. Scramble cube can be regarded as a normal Rubik's cube with $6\times n\times n$ bytes data attached to the surface of it. Since v_j is a rotation tuple (m_a, R, D), which indicates three element of rotation, sub-cube's position shift and sub-square's orientation shift can be done following these three element.

In the fact that every subsquare of the scramble cube is covered by one byte data, at least $4n$ byte data will be scrambled in one round.

4.2.3. Second Phase. At the beginning of the second phase, each scramble cube still has its own six squares, but the bytes on the squares have changed. Then all of the scrambled cubes are unfolded and combined together as a new split cube like Figure 5 shown, which will be split into $6\times n\times n$ fragments in the direction vertical to YOZ coordinate plane. The sub-cube with red color in Figure 6 indicates a typical fragment. At last, every fragment contains one byte from each segment. These fragments will be sent to the randomly chosen servers. The location and order information of fragments will be recorded in the fragment index.

4.3. Data Restoration Process. The restoration process is the opposite way as follows.

Firstly, the second phase is redone in an opposite way to get the split cube. Based on the location and order information of fragments, all of the fragments can be combined together exactly to get the split cube.

Secondly, all of the scramble cubes are restored by dividing the split cube in the direction parallel to YOZ coordinate plane. Then calculate the rotation method table $V_i = \{v_1, v_2, \ldots, v_r\}$ and XOR table $W_{ij} = \{w_1, w_2, \ldots, w_{6\times n\times n}\}$ ($i = \{1, 2, \ldots, N/(6 \times n \times n)\}$, $j = 1, 2, \ldots, r$) using the initial value of logistic map, which is input by user.

Thirdly, carry out the first phase of EPPA in an opposite way from the last segment to the first one. In every round, the reverse rotation method table with reverse direction is represented as follows:

$$V_i^{-1} = \left\{v_1^{-1}, v_2^{-1}, \ldots, v_r^{-1}\right\}. \tag{15}$$

V_i^{-1} is converted from V_i by changing the state transition matrix R and the direction shift function D to the opposite way. For example, R_x in V_i will be converted to $R_{x\text{-anti}}$ in V_i^{-1}.

According to V_i^{-1} and W_{ij}, the reversed process of data scramble and partial encryption can be executed. Then reorganize the scramble cube to segment and do the full encryption again with the previous segment. In this way, a segment is restored.

Finally, all these segments are combined together to restore the original data object.

5. Evaluation

A series of analyses and experiments are designed to analyze the efficiency of EPPA. The experiments are run on a computer with Intel core 2 Duo 2.93 GHz processer and 2 GB RAM. In these experiments, the initial value of logistic map is randomly chosen.

5.1. Safe Threshold. Safe threshold algorithms can be characterized by three parameters (p, m, n), which means in a p-m-n threshold scheme, data object is encoded into n shares such that any m of the shares can reconstruct data object and less than p reveals no information about the encoded data. EPPA, IDA, and secret sharing all belong to safe threshold algorithms.

Both of IDA and secret sharing encode data into n shares and need just m shares to reconstruct the original data object, so attackers have the opportunity to reconstruct data object with m shares. Besides, IDA will breach the confidentiality while the attackers get only one share.

The original data object is split into n shares in EPPA, which can be reconstructed only if the attackers get all of the shares and encryption key. Though data information of each fragment comes from the original data object, it is useless until the attackers reconstruct the whole data object.

5.2. Key Sensibility Analysis and Key Space Analysis. While the size of Rubik's cube is chosen, every segment can be considered as a fixed-length group of bits. Therefore, EPPA can be considered as a block encryption algorithm operating on these groups of bits with an unvarying transformation that is specified by a symmetric key. According to the Strict Avalanche Criterion, a small change in either the key or the plaintext should cause a drastic change in the ciphertext.

Figure 8 shows the percentage of changed cipher text in different rotation round and different size of Rubik's cube, when the initial value of logistic map changes 1×10^{-10}. The results indicate that a small change in the initial value will cause a drastic change in the ciphertext. This result means that when a new initial value of logistic map changes 1×10^{-10}, it can be considered as a new key.

Because the cipher text created by EPPA contributes very little to the key, EPPA can resist statistical attack. According to the result in Figure 8, the key space of EPPA is determined by the following parameters.

5.2.1. Initial Value of Logistic Map. It is a double-precision floating-point format value. Due to the limitation of the accuracy of computer, the computational accuracy of logistic map is 10^{-10} in the experiments.

5.2.2. The Combination Order of Fragments. Suppose that the attacker gets all fragments. Since the number of fragments needed to reconstruct the whole split cubes is $6 \times n \times n$, $n \geq 3$, there will be $(6 \times n \times n)!$ kinds of ways to make all the bytes into cubes.

In conclusion, the total size of key space should be

$$10^{10} \times (6 \times n \times n)!. \tag{16}$$

If $3 \times 3 \times 3$ Rubik's cube is chosen, the size of key space will be $10^{10} \times 54! \approx 2.3 \times 10^{81}$. Correspondingly, while $4 \times 4 \times 4$ Rubik's cube is chosen, the size of key space will be $10^{10} \times 96! \approx 9.9 \times 10^{159}$.

As $(6 \times n \times n)!$ will become extremely huge while n increased, the total size of key space increases sharply while choosing larger cube. As a result, the brute force attack is impossible for EPPA, because even if the attacker uses one thousand servers in the cloud environment and every server executes one billion exhaustive search attacks in a second, it may still cost him 7.3×10^{61} years while $3 \times 3 \times 3$ Rubik's cube is chosen.

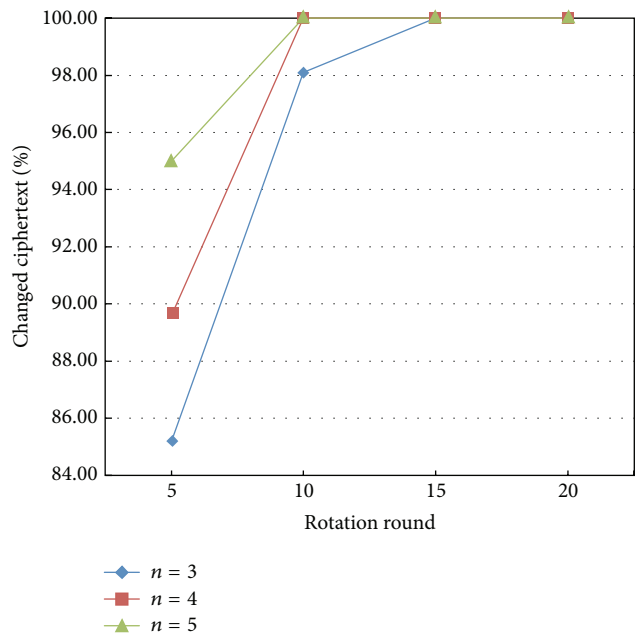

Figure 8: Key sensibility.

5.3. Linear Cryptanalysis and Differential Cryptanalysis. The linear cryptanalysis and the differential cryptanalysis are two essential methods of cryptanalysis. Differential cryptanalysis is up to now one of the most efficient attacks known for iterating cryptography [37]. The differential cryptanalysis belongs to the known plaintext attack. Its basic idea is that we analyze how the difference between a pair of plaintext influences the difference between a pair of cipher to recover certain key bits.

EPPA's differential approximate probability (DP) and linear approximate probability (LP) can be described, respectively, as follows:

$$\begin{aligned} \mathrm{DP} &\leq \frac{\mathrm{DP}_r}{\mathrm{CO} \times \mathrm{RM}}, \\ \mathrm{LP} &\leq \frac{\mathrm{LP}_r}{\mathrm{CO} \times \mathrm{RM}}. \end{aligned} \tag{17}$$

DP_r and LP_r indicate differential approximate probability and linear approximate probability of r round transformation, respectively. CO denotes the combination order of fragments, while RM denotes the scramble cubes' rotation method.

Jakimoski and Kocarev [16] have proposed that the function f's differential approximate probability (DP_f) in a block encryption algorithm is $2^{-5} < \mathrm{DP}_f = 12/256 < 2^{-4}$, and its linear approximate probability (LP_f) is $\mathrm{LP}_f = 2^{-4}$. Reference

[16] has also proven that 11 round transformations' differential (linear) track at least contains 17 chaos sequences in the block encryption algorithm. In EPPA, the round-transformation time is r, which leads to that r round transformation contains $(r \times 17)/11$ chaos sequences at least, so $\mathrm{DP}_r = (2^{-4.678})^{(r\times17)/11}$ and $\mathrm{LP}_r = (2^{-4.678})^{(r\times17)/11}$.

However, the attacker also needs the combination order of the $6 \times n \times n$ fragments and the scramble cubes' rotation method to get the original data information.

Let $N_{\mathrm{SC}|n\times n\times n}$ represent the pattern space of the $n \times n \times n$ scramble cube. For example, as the $3 \times 3 \times 3$ Rubik's cube has eight corners and twelve edges, there are 8! ways to arrange the corner cubes. Seven can be oriented independently, and the orientation of the eighth depends on the preceding seven, giving 3^7 possibilities. There are 12!/2 ways to arrange the edges, since an even permutation of the corners implies an even permutation of the edges as well. Eleven edges can be flipped independently, with the flip of the twelfth depending on the preceding ones, giving 2^{11} possibilities. As a result, the size of pattern space of a $3 \times 3 \times 3$ scramble cube is $N_{\mathrm{SC}|3\times3\times3} = 8! \times 3^7 \times 12!/2 \times 2^{11} \approx 4.3 \times 10^{19}$. Another size scramble cube's pattern space, such as $N_{\mathrm{SC}|4\times4\times4}$, can be figured out in the same way. Since $N_{\mathrm{SC}|n\times n\times n}$ is really large, using exhaustive search attack on the scramble cubes' rotation methods will be the choice when r is small. As one scramble cube has $6 \times n$ rotation methods, $\mathrm{RM} = (6 \times n)^r$.

Since there will be $(6 \times n \times n)!$ kinds of ways to make all the bytes into cubes, $\mathrm{CO} = (6 \times n \times n)!$. As a result, DP and LP can be described, respectively, as follows:

$$\mathrm{DP} \le \frac{\left(2^{-4.678}\right)^{(r\times17)/11}}{(6\times n\times n)! \times (6\times n)^r},$$
$$\mathrm{LP} \le \frac{\left(2^{-4}\right)^{(r\times17)/11}}{(6\times n\times n)! \times (6\times n)^r}. \tag{18}$$

So if $n = 3$ and $r = 10$ are chosen, the results of DP and LP, respectively, are

$$\mathrm{DP} \approx 2^{-351},$$
$$\mathrm{LP} \approx 2^{-340}. \tag{19}$$

Based on Jakimoski and Kocarev's research, it is impossible to construct the differential (linear) characteristic of the block encryption algorithm. This means the differential or the linear cryptanalysis cannot decipher the information generated by this algorithm.

5.4. Memory Cost Analysis. A three-dimensional array is used to represent the subcube's position, so the rotation of cube can work by matrix operations. The value of every element in the array is an object, which contains one-byte data and another byte that used to represent the subsquare's direction. Therefore, only $2 \times 6 \times n \times n$ bytes are needed to represent one scramble cube. If $5 \times 5 \times 5$ scramble cube is chosen, EPPA only needs 300 bytes to represent one scramble cube. Since the iterative calculations also need little memory, EPPA is really cost effective on memory occupation.

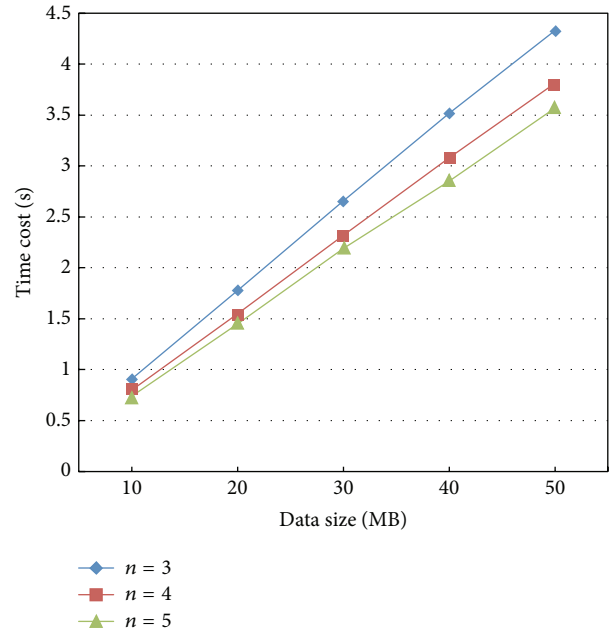

FIGURE 9: Time cost of EPPA under different size of scramble cube while rotation round is $r = 5$.

5.5. Time Cost Analysis

5.5.1. Time Cost in Different Situation. Figure 9 shows the time cost of EPPA under different size data and different size of scramble cube while rotation round $r = 5$. The time cost of EPPA declines, when the bigger size of scramble cube is chosen.

Figure 10 indicates the time cost of EPPA under different size data and different rotation round size while $5 \times 5 \times 5$ scramble cube is chosen. The time cost of EPPA increases along with the rotation round.

These two figures all indicate that EPPA takes liner time to process data object.

5.5.2. Time Cost Compared with AES and DES. Encrypt algorithm library called soft crypto, which is a kind of software cryptographic algorithm library, is used in the experiments. The comparison between EPPA and AES and DES by processing different size of data and calculating the time cost of processing is carried out. These encryption algorithms are widely used.

Figure 11 compares split time cost by EPPA with AES and DES. These experiments choose $5 \times 5 \times 5$ scramble cube with rotation transformation $r = 5$.

Statistics indicate that EPPA with $5 \times 5 \times 5$ scramble cube are faster than AES-128, but slower than DES. AES-128's encryption time is almost 7 times as EPPA ($n = 5, r = 5$). However, the time cost of DES is only 0.75 times as EPPA ($n = 5, r = 5$).

6. Conclusions

EPPA is presented to provide data privacy preserving for cloud storage environments, which can be used as a security

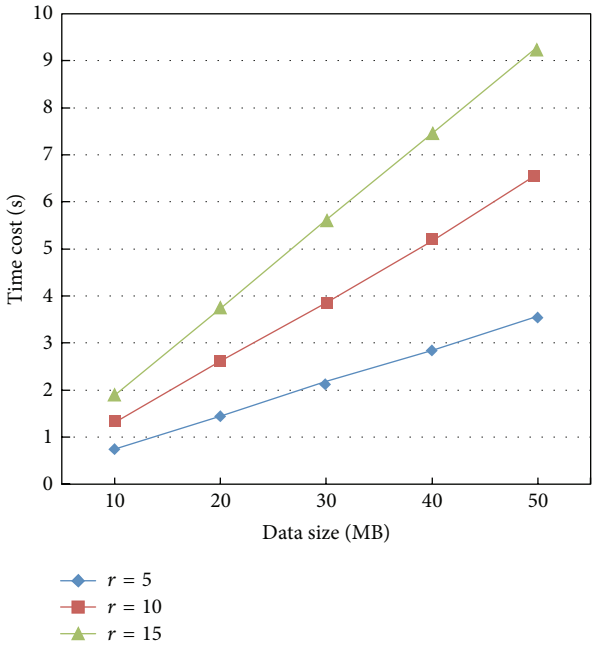

Figure 10: Time cost of EPPA under different rotation round while $5 \times 5 \times 5$ scramble cube is chosen.

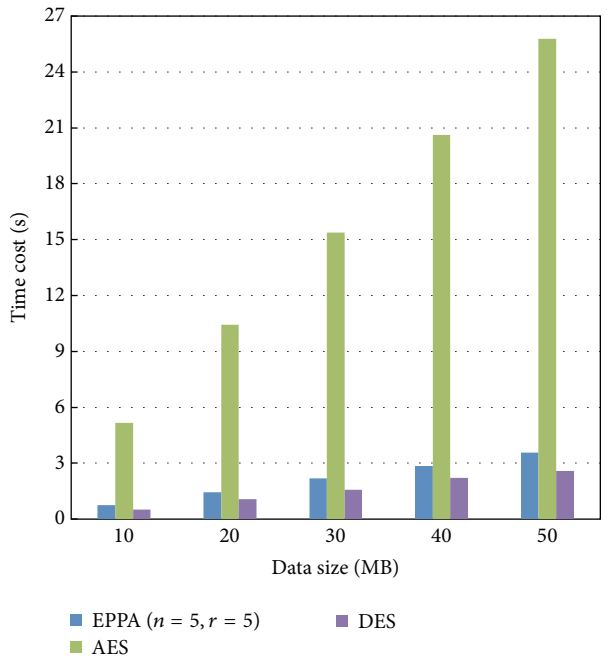

Figure 11: Time cost comparison of EPPA, AES, and DES.

algorithm to encrypt and partition original data object into fragments. EPPA strengthens the confidentiality of data in two phases. Data object is the original data which is encrypted and scrambled at the first phase, and thus the former semantics becomes meaningless. At the second phase, data object is partitioned to break the data locality feature, so that malicious users cannot parse any semantics information even if they obtain some continuous fragments. Analysis proves that the confidentiality of EPPA is useable. Moreover, experiments show that EPPA has decent run-time efficiency.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This study is supported by the National Natural Science Foundation of China (61370069), the National High Technology Research and Development Program ("863" Program) of China (2012AA012600), and the Fundamental Research Funds for the Central Universities (BUPT2011RCZJ16) and China Information Security Special Fund (NDRC).

References

[1] Y. Kim, F. Maino, M. Narasimha, K. H. Rhee, and G. Tsudik, "Secure group key management for storage area networks," *IEEE Communications Magazine*, vol. 41, no. 8, pp. 92–99, 2003.

[2] M. Armbrust, A. Fox, R. Griffith et al., "Above the clouds: a Berkeley view of cloud computing," Tech. Rep., University of California, Berkeley, Calif, USA, 2009.

[3] J. Kubiatowicz, D. Bindel, Y. Chen et al., "Oceanstore: an architecture for global-scale persistent storage," in *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 190–201, ACM Press, Cambridge, Mass, USA, 2000.

[4] A. Adya, W. J. Bolosky, M. Castro et al., "FARSITE: federated, available, and reliable storage for an incompletely trusted environment," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI '02)*, pp. 1–14, Boston, Mass, USA, December 2002.

[5] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," in *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, pp. 188–201, Alberta, Canada, 2001.

[6] E. Gheringer, "Choosing passwords: security and human factors," *Proceedings of IEEE*, vol. 90, pp. 369–373, 2002.

[7] R. W. Proctor, M.-C. Lien, K.-P. L. Vu, E. E. Schultz, and G. Salvendy, "Improving computer security for authentication of users: influence of proactive password restrictions," *Behavior Research Methods, Instruments, and Computers*, vol. 34, no. 2, pp. 163–169, 2002.

[8] S. Riley, "Password security: what users know and what they actually do," *Usability News*, vol. 8, no. 1, 2006.

[9] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *Journal of the ACM*, vol. 36, no. 2, pp. 335–348, 1989.

[10] A. Subbiah and D. M. Blough, "An approach for fault tolerant and secure data storage in collaborative work environments," in *Proceedings of the ACM Workshop on Storage Security and Survivability (StorageSS '05)*, pp. 84–93, November 2005.

[11] A. Shamir, "How to share a secret," *Communications of the Association for Computing Machinery*, vol. 22, no. 11, pp. 612–613, 1979.

[12] G. R. Blakley, "Safeguarding cryptographic key," in *Proceedings of the National Computer Conference*, pp. 313–317, Springer, Montvale, NJ, USA, 1979.

[13] K. Greenan, M. Storer, E. L. Miller, and C. Maltzahn, "POTSHARDS: storing data for the long-term without encryption," in *Proceedings of the 3rd IEEE International Security in Storage Workshop (SISW '05)*, pp. 12–20, San Francisco, Calif, USA, December 2005.

[14] M. Storer, K. Greenan, E. Miller, and K. Voruganti, "POTSHARDSa secure, recoverable, long-term archival storage system," *ACM Transactions on Storage*, vol. 5, no. 2, pp. 1–35, 2009.

[15] L. Kocarev and G. Jakimoski, "Logistic map as a block encryption algorithm," *Physics Letters A*, vol. 289, no. 4-5, pp. 199–206, 2001.

[16] G. Jakimoski and L. Kocarev, "Differential and linear probabilities of a block-encryption cipher," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 1, pp. 121–123, 2003.

[17] G. Jakimoski and L. Kocarev, "Chaos and cryptography: block encryption ciphers based on chaotic maps," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, no. 2, pp. 163–169, 2001.

[18] A. Parakh and S. Kak, "Online data storage using implicit security," *Information Sciences*, vol. 179, no. 19, pp. 3323–3331, 2009.

[19] A. Parakh and S. Kak, "Space efficient secret sharing for implicit data security," *Information Sciences*, vol. 181, no. 2, pp. 335–341, 2011.

[20] A. Parakh and W. Mahoney, "Privacy preserving computations using implicit security," in *Proceedings of the 22nd International Conference on Computer Communication and Networks (ICCCN '13)*, pp. 1–6, August 2013.

[21] Z. Chen, W. Yao, D. Xiao, C. Wu, J. Liu, and C. Wang, "ESSA: an efficient and secure splitting algorithm for distributed storage systems," *China Communications*, vol. 7, no. 4, pp. 89–95, 2010.

[22] M. Löbbing and I. Wegener, "The number of knight's tours equals 33,439,123,484,294–counting with binary decision diagrams," *The Electron Journal of Combinatorics*, vol. 3, no. 1, 1996.

[23] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st annual ACM symposium on Theory of Computing (STOC '09)*, pp. 169–178, ACM, Bethesda, Md, USA, 2009.

[24] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS '12)*, pp. 309–325, January 2012.

[25] C. Gentry, "Computing arbitrary functions of encrypted data," *Communications of the ACM*, vol. 53, no. 3, pp. 97–105, 2010.

[26] M. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT '10)*, pp. 24–43, 2010.

[27] D. Micciancio, "A first glimpse of cryptography's Holy Grail," *Communications of the ACM*, vol. 53, no. 3, p. 96, 2010.

[28] G. Kaddoum, F. Gagnon, and F. Richardson, "Design of a secure Multi-Carrier DCSK system," in *Proceedings of the 9th International Symposium on Wireless Communication Systems (ISWCS '12)*, pp. 964–968, IEEE, Paris, France, August 2012.

[29] G. Kaddoum and F. Gagnon, "Error correction codes for secure chaos-based communication system," in *Proceedings of the 25th Biennial Symposium on Communications (QBSC '10)*, pp. 193–196, May 2010.

[30] Z.-L. Zhu, C. Wang, H. Chai, and H. Yu, "A chaotic image encryption scheme based on magic cube transformation," in *Proceedings of the 4th International Workshop on Chaos-Fractals Theories and Applications (IWCFTA '11)*, 2011.

[31] L. Zhang, X. TiaN, and S. Xia, "Scrambling algorithm of image encryption based on Rubik's cube rotation and logistic sequence," in *Proceedings of the International Conference on Multimedia and Signal Processing (CMSP '11)*, pp. 312–315, Guilin, China, May 2011.

[32] K. Loukhaoukha, J.-Y. Chouinard, and A. Berdai, "A secure image encryption algorithm based on Rubik's cube principle," *Journal of Electrical and Computer Engineering*, vol. 2012, Article ID 173931, 13 pages, 2012.

[33] A.-V. Diaconu and K. Loukhaoukha, "An improved secure image encryption algorithm based on Rubik's cube principle and digital chaotic cipher," *Mathematical Problems in Engineering*, vol. 2013, Article ID 848392, 10 pages, 2013.

[34] R. Matthews, "On the derivation of a "chaotic" encryption algorithm," *Cryptologia*, vol. 13, no. 1, pp. 29–42, 1989.

[35] D. D. Wheeler, "Problems with chaotic cryptosystems," *Cryptologia*, vol. 13, pp. 243–250, 1989.

[36] D. D. Wheeler and R. A. J. Matthews, "Supercomputer investigations of a chaotic encryption algorithm," *Cryptologia*, vol. 15, no. 2, pp. 140–152, 1991.

[37] E. Biham and A. Shamir, *A Differential Cryptanalysis of the Data Encryption Standard*, Springer, Berlin, Germany, 1993.