SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2015; **8**:3771–3781
Published online 24 June 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1298

RESEARCH ARTICLE

Achieving data privacy on hybrid cloud

Xueli Huang and Xiaojiang Du*

Department of Computer and Information Sciences, Temple University, Philadelphia, PA, 19122 U.S.A.

ABSTRACT

In this paper, we propose a novel scheme that can achieve data privacy by hybrid cloud, which consists of public and private cloud, and reduce storage and computation in private cloud, as well as communication overhead between private and public cloud. Meanwhile, we propose a novel algorithm to process private image data. Our experimental results show that (1) our algorithm achieves data privacy but only takes about 1/1000, time of the Advanced Encryption Standard algorithm and (2) the delay of our hybrid cloud approach (including the private and public cloud communications) is only 3%–5% more compared with the traditional public cloud-only approach. Copyright © 2015 John Wiley & Sons, Ltd.

KEYWORDS
data privacy; hybrid cloud; cloud service provide; image

***Correspondence**
Xiaojiang Du, Department of Computer and Information Sciences,Temple University,Philadelphia, PA, 19122 U.S.A.
E-mail: dxj@ieee.org

1. INTRODUCTION

With the rapid development of information and communication technology, the amount of data produced by organizations has grown exponentially, which makes it hard for many organizations to cost-effectively store and manage the data. Cloud computing, a new business model, is considered as one of the most cost-effective solutions for organizations to improve their IT segment and provides the advantage of reduced cost through sharing computing and storage resources. It utilizes an on-demand provisioning mechanism and a pay-per-use model, and has drawn much attention in recent years [1–3].

As more and more individuals and organizations stored their data in cloud, there are also increasing concerns about cloud computing, which have greatly affected the wide adoption of cloud [4–6]. At the top of the list are security and privacy concerns: people concern about the storage and processing of sensitive data in remote physical infrastructures which are owned by a third party, that is, a cloud service provider (CSP). Because a CSP has full control of the physical infrastructure stored clients' data, it is possible that the CSP conducts malicious attacks on users' data for financial or other reasons. For example, a CSP could make money by revealing the data of one client (say C) to C's competitor. Meanwhile, a client's data may be leaked to the public by attackers if a CSP does not have good security mechanisms to protect its servers.

Most existing solutions (e.g., [7,10–13]) employ encryption/decryption techniques combined with access control and auditing system to provide security and privacy for data stored on public cloud. However, in doing so, these solutions inevitably bring in a heavy computational overhead to the data owner for key distribution, data management, data query, and other operations.

In this paper, we consider a different approach: achieving data privacy by utilizing hybrid cloud. A hybrid cloud consists of public cloud (such as Amazon EC2) and private cloud, which is owned and controlled by the data owner. The privacy of data is protected by splitting user data into sensitive data and non-sensitive data and only outsourcing the non-sensitive data to the public cloud. The sensitive data are stored in user's private cloud.

Many data (such as medical data) stored in cloud have a large number of images, which require more storage and computations. A patient medical image may be private. If we directly take advantage of the approach mentioned previously, all the medical images need to be stored in private cloud. This would require a large amount of storage in private cloud and may cause most data stored (and processed) in private cloud, instead of in public cloud. Typically, one wants to minimize the storage and computation in private cloud. To address the aforementioned challenge, an important problem should be addressed: How to efficiently achieve image data privacy by using hybrid cloud? Compared with using public cloud only, using hybrid cloud would have communication overhead between private and public cloud. Besides achieving data privacy, we want to reduce storage and computation in private cloud, as well as communication overhead between private and public cloud.

Another way to protect data privacy is to transform data or perturb data, which are used to preserve the anonymity of data during data dissemination process. Perturbing data is to use techniques like adding noise and swapping values while ensuring that some statistical properties of the entire dataset are maintained. The method of adding noise is used to perturb data before sending it to the public cloud [14], and such noise is removed when these data are sent to authorized users. However, if not enough noise is added, the data can still be recognizable. Hence, this method may introduce a large overhead on communication and processing.

Data transformation is an anonymous method that is used to keep data privacy via generalizations and suppressions. An example of data transformation is to replace the exact date of birth by only the year of birth. The loss of specificity makes the identification process harder [15]. It may not be a good solution to remove any data for privacy purpose because we would not be able to obtain the original information, for example, the month and day of birth.

Our goal is to design a solution that can efficiently provide privacy to data stored in cloud without introducing large overhead on both computation and communication.

In this paper, we propose a novel algorithm that efficiently achieves data privacy for large data sets, especially images, stored in cloud. In our algorithm, firstly, a random noise is added to image blocks instead of pixels, and the size of block is determined by a balance between the complexity of recovering the image and communication overhead. Then a random shuffle operation is applied on the modified blocks, which makes the image hard to be recognized.

The rest of the paper is organized as follows. Section 2 overviews the related work on privacy and security in cloud computing. Section 3 describes our system and thread models. The details of our scheme are presented in Section 4, followed by performance evaluation in Section 5. Finally, our concluding remarks are given in Section 6.

2. RELATED WORK

Much work has been carried out on providing data privacy and security in cloud computing. In the sections that follow, we discuss some closely related work.

2.1. Providing data privacy on virtual machines

Mundada *et al.* [16] introduce SilverLine, a system that enables a CSP to offer security as a service to protect tenant data in clouds even if the software or services that a particular tenant runs are themselves insecure. In SilverLine, the cloud provider's virtual machine manager is augmented to obtain data and network isolation between different cloud users, and these two kinds of isolation are combined with the modifications of guest operating systems to obtain additional data isolation. Data are labeled with security levels by the user, which are used to ensure that data from one user are not propagated to untrusted server instances belonging to other users or to locations outside the cloud. Such kinds of techniques could protect against data leaks that result from compromise, misconfiguration, or side-channel attacks from co-resident cloud tenants [17], but all the protections (including data privacy) are established on the assumption of full trust on the cloud providers.

2.2. Providing data privacy via cryptographic techniques

Feldman *et al.* [7] propose a generic framework – SPORC, for building a wide variety of collaborative applications like word processing and calendaring with untrusted servers. In SPORC, data are encrypted with users' cryptographic keys before being sent to a cloud-hosted server; therefore, the server observes only encrypted data and cannot deviate from the correct execution without being detected. The server assigns a total order to all operations and redistributes the ordered updates to clients, and SPORC clients can detect the server's misbehavior according to the order. Because the order used to detect servers' bad behavior comes from the server, the user is still required to trust the cloud provider to maintain data privacy and even to operate correctly. Furthermore, large computational overhead is introduced because of the use of traditional encryption/decryption.

2.3. Protecting data privacy via hybrid cloud

To secure outsourcing of data and arbitrary computations to an untrusted commodity cloud, in [9], the data are encrypted and verified in private cloud and operated and stored in untrusted public cloud. As we have mentioned previously, much computation overhead will be introduced when the volume is huge, especially for the image data. Zhang *et al.* [8] design a system named Sedic, which leverages the special features of MapReduce to automatically partition a computing job according to the security levels of the data it works on to make the computation with sensitive data in private cloud and only assigns the computation without sensitive data to a public cloud. However, all images contained privacy information will be stored in private cloud, which makes the advantage of hybrid cloud meaningless.

2.4. Access control in cloud

Yu *et al.* [11] propose a scheme, which utilizes and uniquely combines techniques of attributed-based encryption (ABE), proxy re-encryption and lazy re-encryption. Each data file is associated with a set of attributes, and each user is assigned an expressive access structure defined over the attributes of files. Fine-grained access control is received via key-policy ABE (KP-ABE) [18]. Then proxy re-encryption [19] is combined with KP-ABE, which enables the data owner to authorize most computation tasks of user revocation to cloud servers without disclosing the underlying file contents. Finally, the scheme takes advantage of the lazy re-encryption [20] technique to eliminate re-encryptions required as part of access revocation. However, attributed-based encryption is computational expensive and consumes much time of computing resources.

In [13], a completely different approach to secure the data stored in public cloud is proposed via using offensive decoy information technology, which is call Fog computing. Data access is monitored in public cloud, and the abnormal data access is able to be detected through data access patterns. When unauthorized access is suspected and then verified using challenge questions, a disinformation attack is launched by returning large amounts of decoy information to the attacker.

2.5. Achieving data storage security via third party auditor

Users should be able to check the integrity of data placed in cloud, sometimes with the aid of a third party auditor (TPA). Auditing should be efficient and does not require copy of users' data. Auditing should not bring in new vulnerabilities or introduce too much communication and computational overhead. Wang *et al.* [10] propose an auditing protocol by utilizing the technique of public key-based homomorphic authenticator [21–23]. In [10], the TPA could audit data in cloud without a copy of data and thus does not introduce much overhead. Neither can the TPA obtain any knowledge about the data stored in the cloud. Even though the TPA in [10] can protect data from being modified or deleted by the cloud providers, the cloud provider can still access the original data and obtain private information from the data. Hence, the scheme in [10] does not provide data privacy.

3. ACHIEVING DATA PRIVACY VIA HYBRID CLOUD

3.1. System and threat model

The architecture of a hybrid cloud is illustrated in Figure 1. The original data come from private cloud and are sent to data processing center.

For image data containing sensitive information, as seen in Figure 2, we divide the image into several blocks with the same size and shuffle these blocks with a random permutation, and then we transform the problem of image security into a jigsaw problem. To make the jigsaw problem NP-complete, where NP stands for "nondeterministic polynomial time", we add random noise into each color dimension of every block, which breaks the pairwise blocks relationship and the statistic information of the original image. The shuffled and modified image becomes unreadable and is sent to the public cloud. We store the random noise used to cut the relationship between adjacent blocks and the information used to shuffle blocks in the private cloud. The ID of the image is sent to the private and public cloud at the same time. The shuffle order is the information of random permutation obtained from

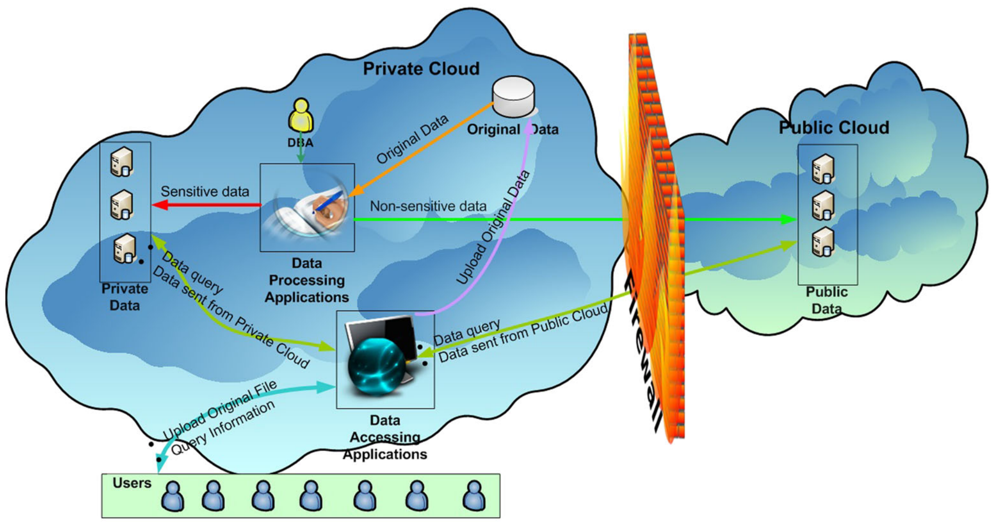

Figure 1. The architecture of hybrid cloud.

the private cloud and is used to reorder the shuffled image downloaded from the public cloud. Then the random noise is obtained from the private cloud, and it is used to recover the original image, as seen in Figure 3.

We consider an untrusted public cloud which intends to obtain sensitive user data and has full control of the hardware, software, and networks in the public cloud.

3.2. Design goals

We want to protect image data privacy stored in public cloud via hybrid cloud. Specifically, we want to design an efficient image encryption algorithm in trusted private cloud, which could make the processed image stored in public cloud unreadable and could store the small amount of data used to encrypt the image in private cloud. It would require too much storage in private cloud if we simply store the entire image with sensitive information in private cloud. Therefore, our design goal is to achieve image data privacy via hybrid cloud and at the same time reduce the following overheads: (1) the amount of data stored in private cloud, (2) the communication overhead between private and public cloud, and (3) the delay introduced by communications between private and public cloud.

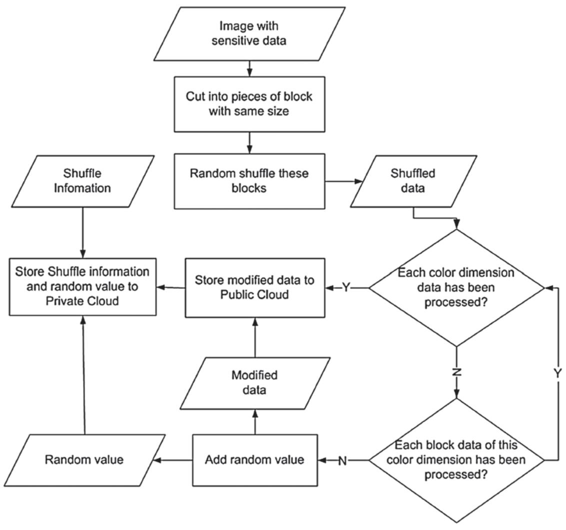
Image with sensitive data
Cut into pieces of block with same size
Random shuffle these blocks
Shuffle Infomation
Shuffled data
Store Shuffle information and random value to Private Cloud
Store modified data to Public Cloud
Y
Each color dimension data has been processed?
N
Y
Modified data
Random value
Add random value
N
Each block data of this color dimension has been processed?

Figure 2. The flowchart of image process.

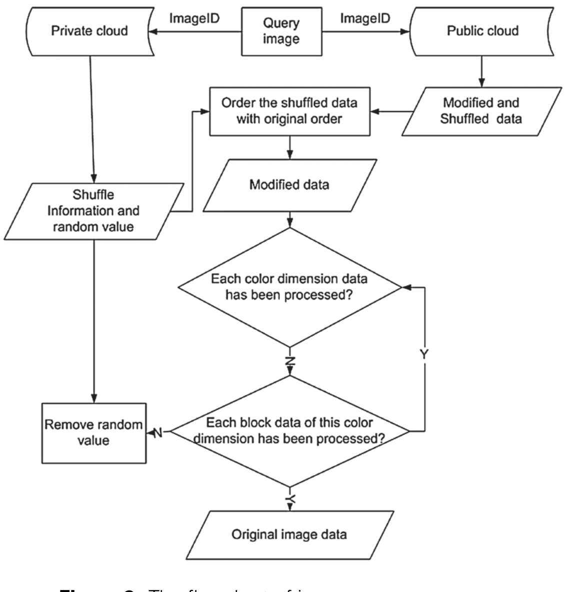
Private cloud
ImageID
Query image
ImageID
Public cloud
Order the shuffled data with original order
Modified and Shuffled data
Modified data
Shuffle Information and random value
Each color dimension data has been processed?
N
Y
Remove random value
N
Each block data of this color dimension has been processed?
Y
Original image data

Figure 3. The flowchart of image recover process.

4. SECURITY OF IMAGE DATA

4.1. Modifying image

4.1.1. Dividing image into blocks.

Different from text, image has a larger size. It is inefficient to perform operations based on pixels, no matter what kind of encryption is taken. To speed up the operation on images, we divide a large image (size of $N \times N$) into n number of blocks, where each block has the same size $k \times k$. Take the Lena image for example, which has a size of 256×256, the size of each block ($k \times k$) is set to 32×32. Therefore, the image is divided into $n = (256 \div 32) \times (256 \div 32) = 64$ pieces, which is shown in Figure 4. The decision of value k, which is related to the degree of security, could be seen in Section 5.1.

4.1.2. Adding noises to each block.

For each color dimension of a block, each pixel value is subtracted from a random value (between 0 and 255). We describe the details of the modification procedure in Algorithm 1. After this modification, features along two adjacent blocks are made unrelated because different random values are used on each color dimension of the two blocks. At the same time, the statistic features of the image is also disarranged, which makes it hard to recover the image via statistic methods.

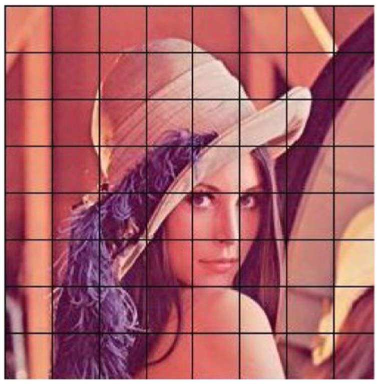

Figure 4. Divided image.

Algorithm 1 Image modification

```
Input: Original image and block size $k$ .
Output: Modified image and a random value image.
for each color dimension do
    for each block of the image do
        $rnd \leftarrow random\ value\ of\ \{0, 1, \ldots, 255\}$;
        for each pixel $p_i$ within the block do
            $tmp \leftarrow p_i - rnd$;
            if $tmp < 0$ then
                $tmp \leftarrow tmp + 256$;
            $p'_i \leftarrow module(tmp, 255)$;
            store $p'_i$ to modified image;
        store $rnd$ to random value image;
```

The random value image, which is used to recover the original image, is stored in the private cloud, and has the size of $\frac{1}{k^2}$ of the original image. Because we randomly choose the value that is used to modify pixels of each color dimension of every block, $(256)^{n \times d3}$ attempts are needed to recover to the original image where $d3$ is the number of color dimensions. Take Lena image, for example, $256^{64\times3}$ attempts are needed. If a 1000 MIPS computer is used, then it will take

$$\frac{256^{64\times3}}{1000 \times 10^6 \times 60 \times 60 \times 24 \times 365} \text{ years}$$

However, the simple way of evaluating the security by counting the number of brute-force trials is not enough to measure the security.

Figure 5 is the Lena image after it is blurred on the block level. Because of human visual capability, the blurred image can still be identified by people who have seen the original one. In addition, people may still be able to gain information from the blurred image even though they have never seen the original image. For example, one can tell that Figure 5 is a photo of girl wearing a hat. To provide strong privacy to image data, we propose to randomly shuffle the blocks of modified image, which is discussed in the next subsection.

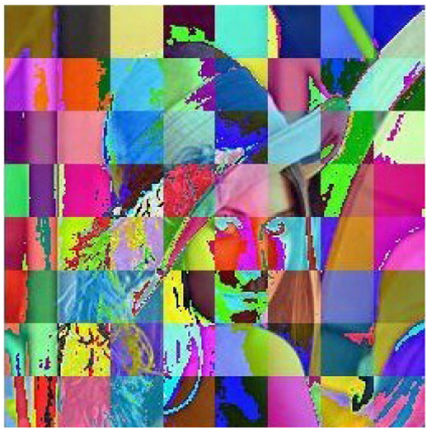
Figure 5. Modified image.

4.2. Random shuffle of blocks

To make the modified image unrecognizable and the storage overhead in private cloud as small as possible, we shuffle the blocks of modified image according to our image shuffle algorithm (the details are given in Algorithm 2). We cluster the n blocks into a number of groups with randomly chosen strides, and each cluster has the same size m.

Algorithm 2 Image shuffle

```
Input: modified image, $n$: the number of block, $m$: the group size.
Output: the shuffled image and shuffle information.
//produce a random permutation $permut$;
for $i = 0 \rightarrow n - 1$ do
    $orig[i] \leftarrow i$;
$left \leftarrow n$;
$cluser\_id \leftarrow 0$;
while $left > 0$ do
    $stride[cluser\_id] \leftarrow$ random value of 1,2,...,n;
    $start[cluser\_id] \leftarrow$ random value of 0,1,...,n-1;/*the start position */
    $i \leftarrow start[cluser\_id]$;
    $i1 \leftarrow 0$;
    for $j = 0 \rightarrow m - 1$ do
        while $orig[i]$ has been chosen do
            $i \leftarrow i + 1$;
            if $i \geq n$ then
                $i \leftarrow i - n$;
        $permut[i1] \leftarrow orig[i]$; /*orig[i]is chosen */
        $i1 \leftarrow i1 + 1$;
        $left \leftarrow left - 1$;
        $i \leftarrow i + stride[cluser\_id]$;
        $cluster\_id \leftarrow cluster\_id + 1$;

//Shuffle modified image with the permutation:$permut$
for $i = 0 \rightarrow n - 1$ do
    $j \leftarrow perm[i]$;
    Copy each pixel within $ith$ block of modified image to $jth$ block of shuffled image;

Store the shuffled image to public cloud;

$shuffle \leftarrow$ each value of $start$ and each value of $stride$ with splitting signal $','$;
Store the string value $shuffle$ to private cloud;
```

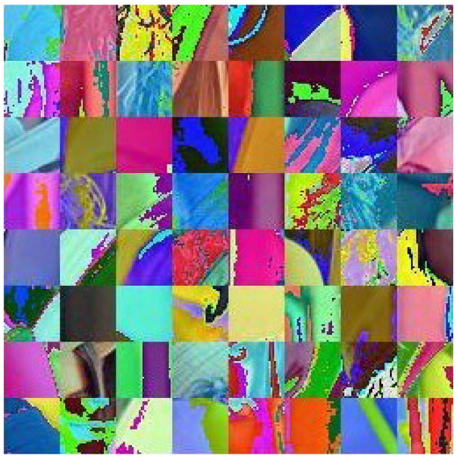

Figure 6. Shuffled image.

The steps are given as follows:

(1) For each cluster of blocks of modified image.

  (a) Randomly choose value *stride* from $1, 2, \ldots, n$ as the stride.
  (b) Randomly choose value *start* from $0, 1, \ldots, n-1$ as the start position.

(2) Increasing the start position, *start* = *start* + *stride*. If the current block has been chosen before, move to the next block (*start* = *start* + 1, until it moves to a block that has not been chosen before. Then copy the block to the shuffled image.
(3) Goto step 2 until m number of blocks have been chosen.
(4) Goto step 1 until all of the blocks of modified image have been chosen.

After modification and shuffle operations are applied on the Lena image, we obtain an image shown in Figure 6, which has become very hard to be identified. This shows that our modification and shuffle approach can protect the privacy of images. In Section 5.1, we will mathematically prove that our approach is secure. The shuffled image is sent and stored in the public cloud, and the shuffle information, such as the random value image, the random stride, and random start position, is stored in the private cloud.

4.3. Recovering images

When an image is queried, the request is sent to both the private cloud and public cloud at the same time. As we mentioned previously, the information used to recover the image is obtained from the private cloud. Firstly, we obtain the shuffle order *permut* via Algorithm 3. Secondly, we re-order the blocks of shuffled image from the public cloud, which makes us obtain the modified image. Thirdly, we obtain the random values from the private cloud and use them to recover the original image from the modified image, that is, the random values are added to each color dimension of every block of the modified image.

Algorithm 3 Obtaining shuffle order

```
Input: $n$: The number of blocks, $shuffle$: a string composed of $start$ value , $stride$ value, and split signal $','$.
Output: Shuffle order $perm$.
//get start, stride, and cluster size information;
$shuff\_string\_array \leftarrow$ split $shuffle$ with $','$;
$start \leftarrow$ the start value of $shuff\_string\_array$;
$stride \leftarrow$ the stride value of $shuff\_string\_array$;
$cluster\_n \leftarrow$ the number of start;
$m \leftarrow \lceil \frac{n}{cluser\_n} \rceil$;
$left \leftarrow n$;
$cluster\_id \leftarrow 0$;
$i1 \leftarrow 0$;
for $i = 0 \rightarrow n-1$ do
  $orig[i] \leftarrow i$;
while $left > 0$ do
  $i \leftarrow start[cluster\_id]$;
  if $m < left$ then
    $m \leftarrow left$;
  for $j = 0 \rightarrow m-1$ do
    while $orig[i]$ has been chosen do
      $i \leftarrow i + 1$;
      if $i \geq n$ then
        $i \leftarrow i - n$;
    $permut[i1] \leftarrow orig[i]$;
    $i1 \leftarrow i1 + 1$;
    $left \leftarrow left - 1$;
  $i \leftarrow i + stride[cluser\_id]$;
  $cluster\_id \leftarrow cluster\_id + 1$;
```

5. PERFORMANCE EVALUATIONS

In this section, we evaluate the performance of our scheme, including security analysis, overhead analysis, and experimental results on efficiency.

5.1. Security analysis

When we divide an image into blocks and then shuffle it, we convert the problem of recovering the shuffled image to original one to the 'jigsaw puzzle problem', which has been proven to be NP-complete if the pairwise affinity among jigsaw pieces is unreliable [29].

Some recent literatures propose to use the features of adjacent block's edge to recover an image such that humans can identify the image content in polynomial time. Cho *et al.* [24] find that the dissimilarity-based compatibility, which is exploited to measure the color difference along the adjacent boundary, is more discriminative than booting-based compatibility [25,26], set-based compatibility [27], and statistics-based compatibility [28].

However, our modifications on an image prevent such features being used to recover a shuffled image. The detailed explanations are given in the succeeding discussion. In our scheme, each pixel of every block's color dimension is modified by subtracting a random value. If after subtraction the pixel is less than 0, the maximum value of pixel (namely 256) is added to the result (to make it positive). Our scheme also destroys the original statistic information. Specifically, our scheme disturbs the statistic features that could be used to obtain the anchor block of a puzzle. This is illustrated in Figures 7 and 8. With the pixel modification in each color dimension of every block, the features of adjacent block edges (used in [24]) are also removed. According to [29], the jigsaw puzzle problem that we have is NP-complete, which means that it cannot be resolved in polynomial time.

We conduct experiments on several color standard test images of size 512×512, and each experiment is run 100 times for each setting. The pairwise affinity is judged by the dissimilarity-based compatibility measurement of the sum of block color difference along adjacent boundaries. Take two blocks blk_i and blk_j, for example, the left–right dissimilarity between them is calculated via Equation (1)

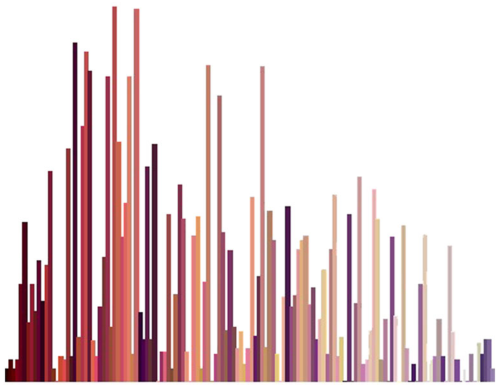

Figure 7. Original image histogram.

$$D(blk_i, blk_j) = \sum_{k=1}^{K} \sum_{l=1}^{d3} ((blk_i(k,u,l) - random(i,l)) - ((blk_j(k,v,l) - random(j,l)))^2 \tag{1}$$

where $d3$ is the number of image color dimensions and each block is a $K \times K \times d3$ matrix, u indexes the last column of blk_i, v indexes the first column of blk_j, and $random(i,k)$ is the array of random values used to modify the original image. The number of blocks n is determined by Equation (2)

$$n = \left\lceil \frac{N \times N}{K \times K} \right\rceil \tag{2}$$

The color difference square D is assumed conform to an exponential distribution, and the probability density function is given in Equation (3).

$$P_{i,j}(blk_j|blk_i) = \lambda e^{-\lambda D(blk_i, blk_j)} \tag{3}$$

where λ is the variance of $D(blk_i, blk_j)$ among all blk_j and $blk_j \neq blk_j$. The sample space Ω is defined as follows:

$$\Omega = \{blk_i | blk_i \ has \ a \ right \ adjacent \ block\} \tag{4}$$

Define event A = 'the block blk_i's right adjacent block has the highest compatibility score among all blocks'. For both the original image and the modified image, the probability that the right adjacent block has the highest compatibility is

$$P(A) = \frac{|A|}{|\Omega|} \tag{5}$$

Figure 9 presents the accuracy probability of identifying block's adjacent block correctly from the shuffled

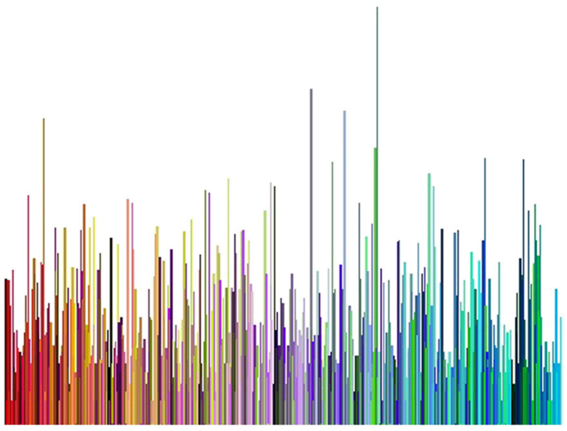

Figure 8. Modified image histogram.

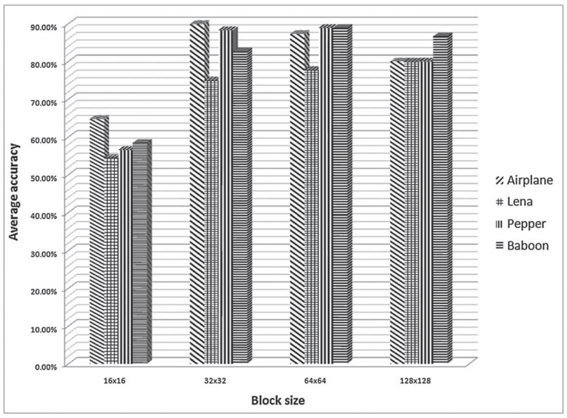

Figure 9. Accuracy without image modification.

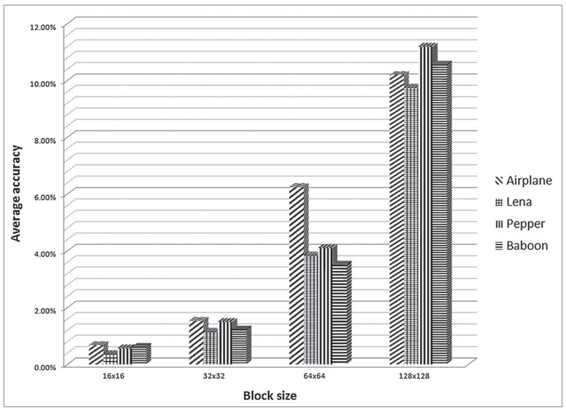

Figure 10. Accuracy after image modification.

original image according to Equation (5), and it can be seen in Figure 9 that the pairwise affinity method is reliable, which means shuffled original image can be recovered in polynomial time.

However, after we modify the original image, the pairwise affinity method does not work well, which can be seen in Figure 10. In Figure 10, the average accuracy to find the right adjacent block from the shuffled modified is very low because of the modification on each block of original image.

We can choose value of K such that the probability in Equation (5) is less than a predefined maximum threshold *maxi_probability*. In Figure 10, we choose $K \le 32$ for the airplane image and $K \le 64$ for Lena, pepper and baboon images, and the average accuracy is low. We can choose other values of K to make the accuracy even lower, which will raise the overhead on both storage and communication. After the maximum of K is determined by *maxi_probability*, the minimum of n is determined according to Equation (2).

For a given shuffled image, suppose it is divided into n pieces and then randomly clustered with size of m, the time to obtain the modified image from the shuffled image on a 1000 MIPS machine is calculated in Equation (6)

$$\frac{n \times n \times (n-m) \times n \times (n-2m) \times n \times \cdots \times 1 \times n}{1000 \times 10^6 \times 60 \times 60 \times 24 \times 365} = \frac{n^{\frac{n}{m}} \times m^{\frac{n}{m}} \times \left(\frac{n}{m}\right)!}{1000 \times 10^6 \times 60 \times 60 \times 24 \times 365} \tag{6}$$

If we set 200 years as a long enough time for image security, take the Lena image (divided into $n = 64$ blocks), for example, we have

$$\frac{n^{\frac{n}{m}} \times m^{\frac{n}{m}} \times \left(\frac{n}{m}\right)!}{1000 \times 10^6 \times 60 \times 60 \times 24 \times 365} > 200 = \frac{64^{\frac{64}{m}} \times m^{\frac{64}{m}} \times \left(\frac{64}{m}\right)!}{1000 \times 10^6 \times 60 \times 60 \times 24 \times 365} > 200 \tag{7}$$

From the aforementioned inequality, the maximum value of m is 8. This means that if we choose $m \le 8$, then it will take more than 200 years for someone to identify the image. This is secure enough for many applications. On the other hand, it is easier to obtain information when an image is divided into less blocks, then the selection of n should be careful. In the previously discussed example, we set n as 64, which is large enough.

Because the noise is randomly chosen for each color dimension of block and the blocks of modified image is shuffled randomly, the information used to modify and shuffle the same image will be different when it is encrypted many times. Therefore, it is secure for the plain text attack.

5.2. Overhead analysis

Achieving data privacy on public cloud is not our only goal; our another goal is to minimize communication overhead introduced by our scheme. Given an image divided into n pieces and randomly clustered with size m, the communication overhead on private cloud is computed in Equation (8)

$$f(m,n) = (n \times n \times 3) + \left(\frac{n}{m} \times 3\right) + c \tag{8}$$

where c is related with the size of TCP/IP header and the image file header. We should minimize the overhead on private cloud, which is formulated as the following optimization problem:

$$\begin{gathered} \text{Minimize } f(m,n) = (n \times n \times 3) + \left(\frac{n}{m} \times 3\right) + c \\ \text{With constraints on :} \\ \frac{n^{\frac{n}{m}} \times m^{\frac{n}{m}} \times \left(\frac{n}{m}\right)!}{1000 \times 10^6 \times 60 \times 60 \times 24 \times 365} > 200 \\ \text{and} \\ P(A) = \frac{|A|}{|\Omega|} < maxi\_probability \end{gathered} \tag{9}$$

As discussed previously, the minimum value of n - *min_n* is determined via inequality $P(A) = \frac{|A|}{|\Omega|} < maxi\_probability$, and the maximum value of m - *max_m* is determined by $\frac{n^{\frac{n}{m}} \times m^{\frac{n}{m}} \times (\frac{n}{m})!}{1000 \times 10^6 \times 60 \times 60 \times 24 \times 365} > 200$. We can obtain all the pairs of (m, n) that satisfy the aforementioned constraints, then we can find the minimum $f(m, n)$.

$$\begin{gathered} min\_overhead = minimum(f(m,n)), \\ where\ 1 \le m \le max\_m\ and \\ n \ge min\_n \end{gathered} \tag{10}$$

5.3. Evaluation of efficiency

To evaluate the efficiency of our privacy preserving method, we compare our algorithm with the Advanced Encryption Standard (AES) algorithm, which is a standard cryptographic algorithm based on permutations and substitutions. We use four different sizes of color Lena image, namely, $128 \times 128, 256 \times 256, 512 \times 512$, and 1024×1024. Both our algorithm and AES (128-bit key) are run on the `Matlab` platform installed in the same computer. For each size of image, we run 100 times modification and recovery operations on the image and obtain the average time. We run AES once to obtain the time of AES encryption and decryption. The results are given in Table I, where the time unit is second.

In Table I, we can see that the time for processing image increases when the image size becomes larger. For all the block sizes, our algorithm is about 1000 times faster than the AES algorithm. The reason is that our algorithm has no iteration, while AES consists of four stages with many rounds. To sum up, our algorithm provides image data privacy, and it is much more efficient than AES.

5.4. Experiments using Amazon EC2

Our private cloud is set up in a server located at Computer Information Systems department of Temple University, and public cloud is built on Amazon EC2 cloud. The Microsoft SQL Server 2005 is one kind of database developed by Microsoft Corp. is installed in the local server, which is used to store private and sensitive data. Amazon Relational Database Service SQL server 2008 is installed in Amazon EC2 and is used to store non-sensitive data. The Microsoft `Visual Studio 2010` software is utilized to create websites that provide services through webpages, which are developed using ASP.NET and C♯.

For database administrator (DBA), we develop data processing applications to provide the process of original data and management of databases both on private cloud and public cloud. And for users, data accessing applications are built to upload the original file and query some kind of information that they are allowed. Internet Information Services is chosen as the web server, which supports both data processing applications and data accessing applications.

The original data, which are uploaded by users via data accessing applications, are stored in private cloud temporarily. The DBA will process the original data using the data processing applications and will output two kinds of data. One is the disturbed data and cannot be recognized, and the other kind of data contains the secret information used to recover the disturbed data. The disturbed data are sent to the public cloud, while the secret information is sent to the private cloud. Meanwhile, the original data will be removed from the original data storage after they are processed by DBA.

When user wants to query some kind of information, a request is sent to the web server via data accessing applications. The web server forwards the request to the private cloud and the public cloud according to the query and user's information. After the web server receives data from private cloud and public cloud, data recovery is performed, and then the recovered data are sent to the user.

As shown in Figure 1, the key information used to recover data is transmitted between data center and web server within the private cloud, and a user is not able to obtain the key information. Meanwhile, only data packets between the web server in private cloud and the data center in public cloud can be obtained at the public cloud. The image data sent to public cloud is either blurred or hashed data, and they cannot be taken advantage of to find connections among tables in public cloud. Therefore, the key information is secured from both the web client and public cloud, which means that our scheme successfully protects data privacy.

We choose the four different sizes of Lena image, namely, $128 \times 128, 256 \times 256, 512 \times 512$ and 1024×1024, to evaluate the security, efficiency, and overhead of our scheme. To measure the delay caused by our scheme, we perform 100 time experiments for each size of Lena image. We record the delay between the time (t1) when a user sends the request and the time (t2) when the web server has the data ready. We also record the delay when our scheme is not used. The average of the 100 runs are reported in Table II, where the time unit is millisecond.

Table II displays that the delay increases as the image size becomes larger, which is easy to understand. Table II also exhibits that our scheme only increases the delay a little bit, between 3.60%–5.29%. This proves the efficiency of our scheme. Meantime, if we compare the data of Table I with that of Table II, we discover that the execution time of our scheme implemented in C♯ (Table II) is much less than that implemented in `Matlab` (Table I, where the time unit is second). The reason is given in the succeeding text: in the C♯ implementation, we take advantage of the LockBitmap

Table I. Running time of our algorithm and AES.

Image size	Our algorithm	AES	Time ratio
128×128	0.1317	122.6	931.1
256×256	0.4593	490.1	1067
512×512	1.858	1957	1054
1024×1024	7.010	7824	1116

AES, Advanced Encryption Standard.

Table II. Comparison of delay.

Image size	With our scheme	Without our scheme	increase
128×128	23.3123	22.5023	3.60%
256×256	76.1976	73.4673	3.72%
512×512	242.4657	230.6346	5.13%
1024×1024	976.4206	927.3578	5.29%

Table III. Overhead on private cloud.

Cluster size	Number of cluster	Communication overhead
1	64	571
2	32	389
3	22	333
4	16	298
5	13	282
6	11	271
7	10	266
8	8	245

Table IV. Overhead on public cloud.

Image size	Communication overhead	Storage overhead
128×128	49499	49206
256×256	197099	196662
512×512	787515	786486
1024×1024	3149171	3145782

class that converts bitmaps to byte arrays, which greatly accelerates the image processing.

We also execute experiments to measure the communication overhead introduced by our scheme. From the overhead analysis in Section 5.2, we know that if image is divided into 64 blocks, the maximum cluster size is 8. The communication overhead of our scheme on private cloud is measured by the packet size, which is captured by the `Wireshark` software installed on the web server. The communication overhead is presented in Table III.

When the number of cluster increases, the communication overhead on private cloud becomes larger. The reason is that the number of random *start* value and *stride* value increases even though the image size is the same.

The overhead of querying Lena image in public cloud is computed using the same method as that on private cloud, and the results are reported in Table IV. Table IV shows that the overhead increases when the image size becomes larger. It is because the maximum transmission unit (MTU) limitation over Internet, which divides larger size data into pieces with size less than MTU. When the image size increases, the image is divided into more pieces and hence has more overhead.

In Tables III and IV, we can calculate the overhead caused by our privacy preserving scheme: (1) If we do not run the optimum value of cluster numbers, the overhead is between $\frac{245}{3149171}$ and $\frac{571}{49499}$, that is, 0.01%–1.52%. (2) Otherwise, the overhead is even less $\frac{245}{49499}$, that is, 0.49%. Therefore, our scheme introduces little overhead.

6. CONCLUSION

To address the increasing concern of data privacy in cloud, we proposed a novel scheme that can protect data privacy, especially for image data. In our scheme, an image containing privacy information is divided into blocks, and the blocks are shuffled with random start position and random stride. Our scheme operates at the block level instead of the pixel level, which greatly speeds up the computation. We converted the image privacy problem into the jigsaw puzzle problem. To make the jigsaw puzzle problem NP-complete, we modified the image data based on blocks by subtracting a random value for each pixel within the same block and same color dimension. These operations make the pairwise affinity unreliable and make the shuffled image unrecognizable as well as the statistic information. We formulated an optimization problem to minimize the overhead. By carefully selecting the number of blocks and the cluster size, the communication overhead of our scheme on private cloud can be greatly reduced. We implemented our scheme in real network environments (including the Amazon EC2) and tested the security, efficiency, and communication overhead. Both our analysis and experimental results showed that our scheme is secure, efficient, and introduces little overhead.

ACKNOWLEDGEMENTS

This work was supported in part by the US National Science Foundation under grants CNS-0963578, CNS-1022552, and CNS-1065444, as well as US Army Research Office under grant WF911NF-14-1-0518.

REFERENCES

1. Mell P, Grance T. *Draft NIST working definition of cloud computing*, 2009. Referenced on June. 3rd.
2. Chen D, Zhao H. Data security and privacy protection issues in cloud computing, *2012 International Conference on Computer Science and Electronics Engineering (ICCSEE)*, Delhi, India, 2012.
3. di Vimercati SDC, Foresti S, Samarati P. Managing and accessing data in the cloud: privacy risks and approaches, *Proceedings of CRiSIS*, Cork, Ireland, 2012.
4. Ryan MD. Cloud computing privacy concerns on our doorstep. *Communications of the ACM* 2011; **54** (1): 36-38.
5. Subashini S, Kavitha V. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications* 2011; **34** (1): 1-11.
6. Jansen WA. Cloud hooks: security and privacy issues in cloud computing, *2011 44th Hawaii International Conference on System Sciences (HICSS)*, Kauai, HI, USA, 2011.
7. Feldman AJ, Zeller WP, Freedman MJ, Felten EW. SPORC: Group collaboration using untrusted cloud resources, *OSDI*, Vancouver, BC, Canada, 2010.
8. Zhang K, Zhou X, Chen Y, Wang X, Ruan Y. Sedic: privacy-aware data intensive computing on hybrid

clouds, *Proceedings of the 18th ACM conference on Computer and communications security,* ACM, Chicago, IL, USA, 2011.

9. Bugiel S, Nürnberger S, Sadeghi AR, Schneider T. Twin clouds: an architecture for secure cloud computing, *Workshop on Cryptography and Security in Clouds, WCSC*, Zurich, Switzerland, 2011.
10. Wang C, Wang Q, Ren K, Lou W. Privacy-preserving public auditing for data storage security in cloud computing, *2010 Proceedings IEEE INFOCOM*, San Diego, CA, USA, 2010.
11. Yu S, Wang C, Ren K, Lou W. Achieving secure, scalable, and fine-grained data access control in cloud computing, *2010 Proceedings IEEE INFOCOM*, San Diego, CA, USA, 2010.
12. Li J, Jia C, Li J, Liu Z. Novel framework for outsourcing and sharing searchable encrypted data on hybrid cloud, *2012 4th International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, Bucharest, Romania, 2012.
13. Stolfo SJ, Salem MB, Keromytis AD. Fog computing: mitigating insider data theft attacks in the cloud, *2012 IEEE Symposium on Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, 2012.
14. Hwang K, Li D. Trusted cloud computing with secure resources and data coloring. *IEEE Internet Computing* 2010; **14**(5): 14-22.
15. Iyengar VS. Transforming data to satisfy privacy constraints, *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, AB, Canada, 2002.
16. Mundada Y, Ramachandran A, Feamster N. Silver-Line: data and network isolation for cloud services, *2nd USENIX Workshop on Hot Topics in Cloud Computing*, Portland, OR, USA, 2011.
17. Ristenpart T, Tromer E, Shacham H, Savage S. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds, *Proceedings of the 16th ACM Conference on Computer and Communications Security*, Chicago, IL, USA, 2009.
18. Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data, *Proceedings of the 13th ACM Conference on CCS*, Alexandria, VA, USA, 2006.
19. Blaze M, Bleumer G, Strauss M. Divertible protocols and atomic proxy cryptography, *Advances in Cryptology EUROCRYPT'98*, Helsinki, Finland, 1998.
20. Kallahalla M, Riedel E, Swaminathan R, Wang Q, Fu K, *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, San Antonio, TX, USA, 2003.
21. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D. Provable data possession at untrusted stores, *Proceedings of the 14th ACM Conference on Computer and Communications Security*, Alexandria, VA, USA, 2007.
22. Shacham H, Waters B. Compact proofs of retrievability, *Advances in Cryptology–ASIACRYPT 2008*, Melbourne, Australia, 2008.
23. Wang Q, Wang C, Li J, Ren K, Lou W. Enabling public verifiability and data dynamics for storage security in cloud computing, *Computer Security–ESORICS 2009*, Saint Malo, France, 2009.
24. Cho TS, Avidan S, Freeman WT. A probabilistic image jigsaw puzzle solver, *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, USA, 2010.
25. Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors. *The Annals of Statistics* 2000; **28**(2): 337-407.
26. Torralba A, Murphy KP, Freeman WT. Sharing features: efficient boosting procedures for multiclass object detection, *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004*, Washington, DC, USA, 2004.
27. Simakov D, Caspi Y, Shechtman E, Irani M. Summarizing visual data using bidirectional similarity, *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR '08*, Anchorage, AK, USA, 2008.
28. Weiss Y, Freeman WT. What makes a good model of natural images? *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07*, Minneapolis, MN, USA, 2007.
29. Demaine ED, Demaine ML. Jigsaw puzzles, edge matching, and polyomino packing: connections and complexity. *Graphs and Combinatorics* 2007; **23**(1): 195–208.