Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 402890, 14 pages
doi:10.1155/2012/402890

Research Article

Re-Encryption Method Designed by Row Complete Matrix

I-Te Chen,[1] Jer-Min Tsai,[2] and Jengnan Tzeng[3]

[1] *Kaohsiung Medical University, Kaohsiung 80708, Taiwan*
[2] *Kun Shan University, Yung-Kang 71003, Taiwan*
[3] *National Chengchi University, Taipei 11605, Taiwan*

Correspondence should be addressed to Jengnan Tzeng, jengnan@gmail.com

Received 20 January 2012; Revised 30 March 2012; Accepted 30 March 2012

Academic Editor: Zheng-Guang Wu

With the prevalence of Internet access, document storage has become a fundamental web service in recent years. One important topic is how to design a secure channel for efficiently sharing documents with another receiver. In this paper, we demonstrate a re-encryption method that is designed with row complete matrices. With this new method, the document owner can share a ciphertext in the cloud with another receiver by sending a serial number to the server and giving the receiver a corresponding key at the same time. This method ensures that the server cannot obtain the information about key and plaintext and that the receiver cannot obtain the original key of the owner either. Only the owner has the knowledge of all the information. Using this re-encryption system, the cloud server can provide a secure file-sharing service without worrying about the shared key management problem. Moreover, the cost of re-encryption will not increase even when the encryption is strengthened with longer encryption keys.

1. Introduction

The latin matrix is an n-by-n square matrix filled with elements from Z_n, each occurring exactly once in each row and in each column. The orthogonal Latin matrices have been found to be useful in error-correcting codes [1] and in mathematical puzzles, SUDOKU. Nevertheless, Colbourn proved that if a partially filled square can be completed to form, a Latin square is NP-complete [2]. In addition, the Latin matrices are the multipermutation method, an important method in cryptography. The Latin matrices include a many-internal state that may be keyed, and combining operations using keyed matrices will be nonlinear. Jiejun Kong shows the role of Latin matrices in Cipher Systems [3].

Blaze et al. introduced the proxy re-encryption (PRE) based on Elgamal encryption in 1998 [4], in which the proxy can translate ciphertexts under Alice's key into ciphertexts under Bob's key with a given re-encryption key. However, traditional PRE can translate all Alice's ciphertexts without any condition. Therefore, Weng et al. proposed conditional proxy re-encryption in 2009 [5]. In Weng's scheme, the proxy can only translate ciphertexts under certain condition. Unfortunately, Weng's scheme was broken and an efficient conditional PRE scheme with bilinear pairing was introduced in 2011 [6]. Another related topic is PRE with keyword search, in which proxy is given a search keyword to find whether or not a ciphertext contains the search keyword without knowing the corresponding plaintext [7, 8].

In 2007, Green and Ateniese [9] presented the first identity-based proxy re-encryption scheme, where ciphertexts are transformed from one identity to another. Moreover, this scheme does not require any extra work from the key generator of a trusted party. Later, Chu and Tzeng [10] also proposed an identity-based proxy re-encryption scheme, which is efficient in terms of both computation and length of ciphertexts. However, the data owners in those schemes above cannot prevent Bob from decrypting the data and generating new re-decryption keys to another user. Hence, Liu et al. proposed a time-based re-encryption scheme, which enables the cloud servers to re-encrypt data based on specified interval [11]. Moreover, Liu's scheme is an attribute-based encryption that provides fine grained access control as each user is issued keys associated with attributes and corresponding attribute effective times. The data can be decrypted by Bob using the attribute keys satisfying the access control and attribute effective times satisfying the access time. Fang et al. combined conditional PRE and attribute-based encryption techniques and proposed interactive conditional proxy re-encryption with fine grain policy (ICPRE-FG) [12] in 2011. In ICPRE-FG system, each ciphertext labels a set of descriptive conditions; each proxy re-encryption key is associated with a tree-access structure where the leaves are associated with conditions. Hence, this structure determines which type of ciphertexts the key can re-encrypt.

In addition, Ateniese et al. [13] proposed the first key-private PRE scheme and also left an open problem about how to achieve key privacy without compromising security against the chosen-ciphertext attack (CCA). Furthermore, Shao et al. proposed the "achieving key privacy without losing CCA security in proxy re-encryption" in 2011 [14] to achieve CCA security. In recent years, as the adoption of cloud computing environment grows, the PRE gradually presents its vital role in cloud computing. Take personal health record (PHR) for example; a PHR contains not only medical history such as surgery, illness, laboratory test results, and imaging report immunization records, but also sensitive personal data such as name, birth date, weight, and contact information. Obviously, such sensitive data should be protected by one's key, and the medical history might be re-encrypted for public health issue in cloud computing environment.

In a cloud computing environment, users may want to exchange their digital contents with other users in the domain of another cloud. However, before doing that, users need to authenticate each other first and exchange a session key for communication. And PRE scheme can serve as a useful tool for content exchange in such cloud computing scenario.

Since the PRE scheme runs the risk of divulging the key of content provider, we will propose a secure re-encryption method designed by applying the properties of latin square matrix. Before we describe this new method, we need to introduce a special type of latin matrix, which is called row complete matrix.

Row Complete Matrix

A row complete matrix is a latin matrix in which, for all pairs $(i, j) \in Z_n^2$ where $i \neq j$, the pair exists in some row of the latin matrix. It is known that when n is 3 and 5, there is no latin matrix that is row complete. It remains an open problem to decide whether or not the row complete matrix exists when n is odd.

For example, the smallest row complete matrix is

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{1.1}$$

This simple case shows that all the pairs $(i, j) \in Z_2$ can be found in the matrix, that is, $(0, 1)$ in the first row and $(1, 0)$ in the second row. The second example is in the case of $n = 4$. For example,

$$\begin{pmatrix} 3 & 2 & 0 & 1 \\ 2 & 1 & 3 & 0 \\ 1 & 0 & 2 & 3 \\ 0 & 3 & 1 & 2 \end{pmatrix}. \tag{1.2}$$

In this example, we see that the pairs $(0, 1)$, $(0, 2)$, and $(0, 3)$ can be found in the first, third, and fourth rows, respectively. Similarly, all the pairs $(i, j) \in Z_4$ for $i \neq j$ can be found in this latin matrix, and therefore this matrix is row complete.

There are only 2 row complete matrices for $n = 2$, 144 row complete matrices for $n = 4$ and 172800 row complete matrices for $n = 6$. In fact, the number of row complete matrices grows rapidly as the even number n increases. We will design a re-encryption method using row complete matrix, and the security will rely on the number of possible row complete matrices for given size n. We will describe some properties of row complete matrix in the following sections.

The remainder of this paper is organized as follows. Section 2 gives the definition of re-encryption and a new re-encryption method designed by the properties of row complete matrix. The detailed procedure of the proposed method and the security analysis are described in Section 3. Finally, Section 4 gives the conclusions.

2. Re-Encryption Method

First, we define the concept of re-encryption. Assume that Alice (A) has some contents that she will upload to the Cloud (C) and Bob (B) is Alice's friend with whom Alice will share her contents through the server in the cloud. Before Alice uploads her contents (P) with the server, she will encrypt her contents with a key, say K_a. Then the ciphertext (T_a) is uploaded to the server. When Alice wants to share her contents (P) to Bob, she will create two keys. The first key is K_b, which is a key that Bob can use to decrypt the ciphertext from the server. The second key is $m_{a,b}$, which is a serial number and with which the server will re-encrypt the ciphertext (T_a). Let the operation of re-encryption be R. We denote the re-encrypted ciphertext by T_b, which is equal to $R(T_a, m_{a,b})$. If we denote the encryption method and the decryption method by E and D, respectively, then we have the following relations:

(i) $T_a = E(P, K_a)$,

(ii) $P = D(T_a, K_a)$,

(iii) $T_b = R(T_a, m_{a,b})$,

(iv) $P = D(T_b, K_b)$.

For the reason of security and feasibility, we set the following criteria. (1) It is difficult for the server, S, to recover the K_a and P from the information of T_a and $m_{a,b}$. (2) It is difficult for Bob, B, to recover K_a from the information of P, T_b, and K_b. Moreover, (3) for the feasibility of cloud transition, the length of the serial number $m_{a,b}$ should be small and the computational cost of $R(T_a, m_{a,b})$ must also be small.

Encryption Method

The row complete matrix plays a pivotal role in our method. In fact, the adjacent column pairs of the row complete matrix serve collectively as the code book in an encryption. We first show the central idea of our method. Based on this idea, we will implement the encryption process to make it more secure.

We begin with the following example. Assuming the key K_a is an n-by-n row complete matrix, we will express the plaintext with the n-base representation first. For example,

$$K_a = \begin{pmatrix} 3 & 2 & 0 & 1 \\ 2 & 1 & 3 & 0 \\ 1 & 0 & 2 & 3 \\ 0 & 3 & 1 & 2 \end{pmatrix}, \tag{2.1}$$

where the size of K_a is 4-by-4. Then, we express the plaintext with the 4-base representation. The first element of the 4-base plaintext is encrypted by the first two columns. If the first element is 3, then it is encrypted by 2, and if it is 2, then it is encrypted by 1. The second element of the plaintext is encrypted by the second and third columns. If the second element is 2, then it is encrypted by 0, and if it is 1, then it is encrypted by 3. If K_a is a 4-by-4 matrix, only three pairs of columns can be used. We will repeat these three pairs when the length of plaintext is greater than 4. The formal encryption process is as follows.

If the key K_a is an n-by-n matrix, the kth element of the n-base representation of the plaintext $P(k)$ is encrypted by $K_a(s, t+1)$, where $K_a(s,t)$ is the element of s row and t column of K_a, $t = \text{mod}(k-1, n-1) + 1$ and s is the index of the tth column of K_a such that $K_a(s,t) = P(k)$.

Decryption Method

The decryption process is straightforward when we have key K_a. Because $E(P(k)) = K_a(s, t+1)$ when $T(k) = K_a(s,t)$, $P(k)$ is decoded by the index t, which is determined by the location index k and the index of the $(t+1)$th column of K_a, where $T(k)$ occurs. That is $P(k) = K_a(s,t)$, where $t = \text{mod}(k-1, n-1) + 1$, and s is the index of the $(t+1)$th column of K_a such that $K_a(s, t+1) = T(k)$.

Before we further explain the re-encryption method, we will first show some important properties for row complete matrix.

Theorem 2.1. *Let A be a row complete matrix; then a row permutation of A also gives a row complete matrix.*

Proof. If A is a row complete matrix, then A is a latin matrix. A latin matrix after a row permutation is still a latin matrix, and the row permutation will not affect the row (i, j) pair. That is, if B is a row permutation from A, the pair (i, j) in the sth row of A must exist in some row of B. Hence, a row complete matrix after a row permutation is also a row complete matrix. □

Corollary 2.2. *Let A be a row complete matrix. If B is a matrix derived from A by row permutation, then A and B are equivalent in the sense that they encrypt the plaintext to the same ciphertext.*

Theorem 2.3. *Let A be a row complete matrix; then permuting the elements of A will again produce a row complete matrix.*

Proof. Let σ be a permutation defined on Z_n and B a matrix such that $B(i, j) = \sigma(A(i, j))$. Since A is a latin matrix, each row and each column contain each element of Z_n exactly once. As σ is a permutation, each row and each column of B contain each element of Z_n exactly once as well; therefore B is also a latin matrix.

Since σ is a permutation mapping, we have $\sigma^{-1}(i) \neq \sigma^{-1}(j)$ if $i, j \in Z_n$ and $i \neq j$. Then for all pairs $(i, j) \in Z_n^2$, $i \neq j$, we have $i' = \sigma^{-1}(i)$, $j' = \sigma^{-1}(j)$, and $i' \neq j'$. Because A is a row complete, there exist some $s \in \{1, \ldots, n\}$ and $t \in \{1, \ldots, n-1\}$ such that $A(s, t) = i'$ and $A(s, t+1) = j'$. Then, with the same indices s and t, we will have $B(s, t) = \sigma(A(s, t)) = \sigma(i') = i$ and $B(s, t+1) = j$. Hence, B is a row complete matrix too. □

From these two theorems, we know that if A is a row complete matrix with size n, there are more than $n!$ row complete matrices that can be derived from the row permutations of A and the element permutations of A. If B is a row complete matrix derived from the row permutation of another row complete matrix A, we can see that using these two matrices as the keys in the previous simple encryption process, we will obtain the same ciphertext. As this property will downgrade the encryption security, we will modify the encryption method to a secure version.

The main idea of the modification is to make the ciphertexts encrypted by A and B that are derived from the row permutation from A different. The number of the first column can be used. If we use the number of the first column (except 0) as the column order for encryption, we can make the above A and B have different ciphertexts. For example, if K_a is as (2.1), then we use the third column to encrypt the first element of plaintext, the second column to encrypt the second element, the first column to encrypt the third element, and then repeat.

The encryption formula is modified as $E(P(k)) = K_a(s, t+1)$, where $t = k_a(\mathrm{mod}(k-1, n-1)+1)$, k_a is the first column of K_a from which the 0 element is removed, and s is the index of the tth column of K_a such that $K_a(s, t) = P(k)$. Similarly, the decryption process is modified as $D(T(k)) = K_a(s, t)$, where $t = k_a(\mathrm{mod}(k-1, n-1)+1)$, k_a is the first column of K_a from which the 0 element is removed, and s is the index of the $(t+1)$th column of K_a such that $K_a(s, t+1) = T(k)$.

Re-Encryption Method

Let K_a be the key of Alice, P the plaintext, and T_a the ciphertext of P that is encrypted by K_a. After encryption, T_a is transmitted to the cloud. If Alice wants to share the document T_a with Bob, Alice will randomly generate a key, say K_b, and then send this key to Bob. At the same time, Alice has to send a re-encryption key $m_{a,b}$ to the cloud server which can produce T_b for Bob. Note that the re-encryption keys $m_{a,b}$ and K_b are one time keys, so the server will not keep the key and do any key management.

The main idea of re-encryption is to create a ciphertext T_b from which Bob can successfully reconstruct the plaintext by K_b. For the ease of understanding, we will use the following example to illustrate. Let the keys of Alice and Bob be

$$K_a = \begin{pmatrix} 2 & 1 & 3 & 0 \\ 3 & 2 & 0 & 1 \\ 1 & 0 & 2 & 3 \\ 0 & 3 & 1 & 2 \end{pmatrix}, \qquad K_b = \begin{pmatrix} 1 & 0 & 2 & 3 \\ 2 & 1 & 3 & 0 \\ 3 & 2 & 0 & 1 \\ 0 & 3 & 1 & 2 \end{pmatrix}. \tag{2.2}$$

We can see that Alice encrypts the plaintext by the order $(2,3,1,2,3,1,\ldots)$ and Bob decrypts the ciphertext by the order $(1,2,3,1,2,3,\ldots)$. The first element of $T_a(T_b)$ will be decoded by the first column of $K_a(K_b)$. If $P(1) = 0$, then $T_a(1) = 2$. If we want $T_b(1)$ to be decoded to $P(1) = 0$, $T_b(1)$ should be 3. However, the difference between $T_a(1)$ and $T_b(1)$ is the message that Alice should provide to the server. If we check all the possible values of $P(1)$, we can see that the difference between $T_a(1)$ and $T_b(1)$ is the same in the sense of mod 4, that is $T_b(1) = \bmod\ (T_a(1)+1, 4)$ for all $P(1)$. Similarly, we can check that $T_b(2) = \bmod\ (T_a(2)+1, 4)$ and $T_b(3) = \bmod(T_a(3) + 2, 4)$. Therefore, the message that Alice should provide to the server is a $n-1$-dimensional vector $(1,1,2)$. When the sever (S) receives this message, it only has to add these values by repeating the order onto T_a and then take mod 4 to obtain T_b.

Note that the message that Alice provides to the server is a vector, not a matrix, and each value of this vector belongs to Z_4. This property satisfies the third criterion; namely, the size of the message for re-encryption is small and the computational cost is also small. However, this advantage does not always hold true for all row complete matrices. For example, let

$$K_a = \begin{pmatrix} 2 & 1 & 0 & 3 \\ 0 & 2 & 3 & 1 \\ 1 & 3 & 2 & 0 \\ 3 & 0 & 1 & 2 \end{pmatrix}, \qquad K_b = \begin{pmatrix} 3 & 0 & 2 & 1 \\ 2 & 3 & 1 & 0 \\ 1 & 2 & 0 & 3 \\ 0 & 1 & 3 & 2 \end{pmatrix}, \tag{2.3}$$

the re-encryption key should be

$$m_{a,b} = \begin{pmatrix} 0 & 2 & 0 & 2 \\ 1 & 0 & 0 & 0 \\ 2 & 2 & 3 & 3 \\ 3 & 0 & 1 & 3 \end{pmatrix}. \tag{2.4}$$

The first column specifies the values appearing in ciphertext. If the kth element of ciphertext is j, then it will be re-encrypted by adding the value in the $(j+1, \operatorname{mod}(k-1, n-1)+2)$ element of $m_{a,b}$. Although, the first column of $m_{a,b}$ is redundant, this re-encryption key is still of a matrix form. Fortunately, there exists large amount of row complete matrices that make the column (except the first column) of $m_{a,b}$ a unique constant so that we can easily transmit the re-encryption key to the server. The following theorem will give us a simple method to derive a new row complete matrix from a given row complete matrix such that its re-encryption key is an $(n-1)$-dimensional vector.

Theorem 2.4. *Let A be a given row complete matrix of size n. There exists a nonzero $(n-1)$-dimensional vector such that when each element of the ith column of A is added by the ith number of the vector in the sense of modular addition, it will become the other row complete matrix.*

Proof. Let the nonzero $(n-1)$-dimensional vector be a constant vector with the constant c. We can see that adding c to every element of A is equivalent to a permutation σ such that $\sigma(i) = \operatorname{mod}(i+c, n)$. By Theorem 2.3, we know that the new matrix $A+c$ is also a row complete matrix. □

What we are interested in is the case of the nonconstant vector. In particular, we would like to decide under what condition the row complete matrix modified by an non-constant vector will become the other row complete matrix. Note that the first flowchart in Figure 1 can quickly generate a row complete matrix although we also observe that it can only produce a subset of all the row complete matrices. Once a row complete matrix is obtained from the first flowchart, we can then efficiently produce another row complete matrix by adding a non-constant vector.

The first flowchart started by the special Latin matrix that is called the simple Latin matrix. The (i, j) element of simple Latin matrix in which each element is defined by $\operatorname{mod}(i+j-2, n)$. Given this simple Latin matrix, we can randomly permute the columns of the matrix until this matrix becomes a row complete matrix. When we obtain the row complete matrix, we randomly permute its rows once again. Up to now, we obtain a row complete matrix. The advantage of this flowchart is that we do not have to check whether the matrix is a Latin matrix because the row and column permutation operation preserves itself as a Latin matrix.

Given two arbitrary row complete matrices, the re-encryption key $m_{a,b}$ related to these two row complete matrices does not always have the same value for each column. Hence, we must have an algorithm to produce a row complete matrix from the others such that the re-encryption key can be reduced to the vector type. The flowchart in Figure 2 describes a simple procedure for the above-mentioned purpose. Since the output row complete matrix is derived by the column permutation of a row complete matrix, the next theorem will show that such re-encryption key can be reduced to a vector form.

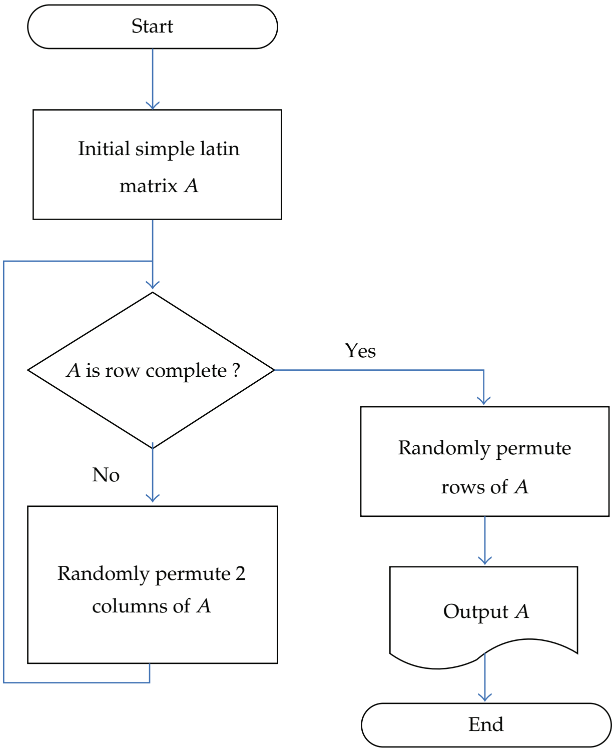
Start
Initial simple latin matrix A
A is row complete ?
Yes
No
Randomly permute rows of A
Randomly permute 2 columns of A
Output A
End

Figure 1: The flowchart to generate a row complete matrix by simple Latin matrix.

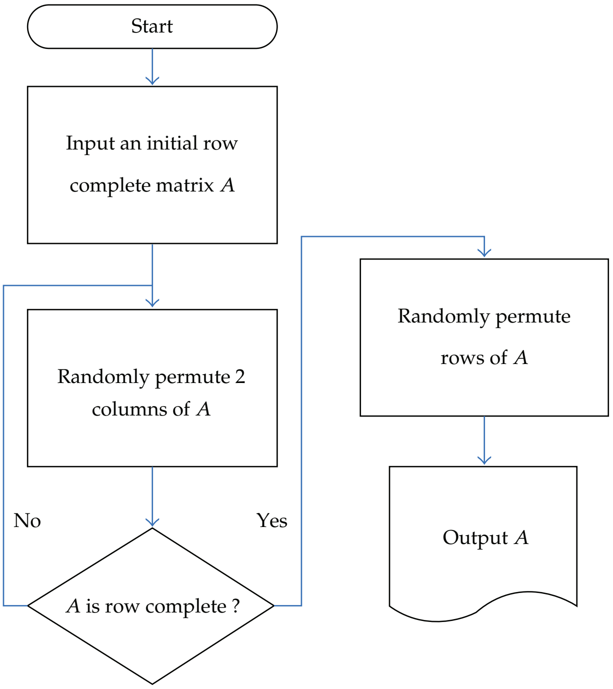
Start
Input an initial row complete matrix A
Randomly permute 2 columns of A
Randomly permute rows of A
No
Yes
Output A
A is row complete ?

Figure 2: The flowchart to generate another row complete matrix by a given row complete matrix.

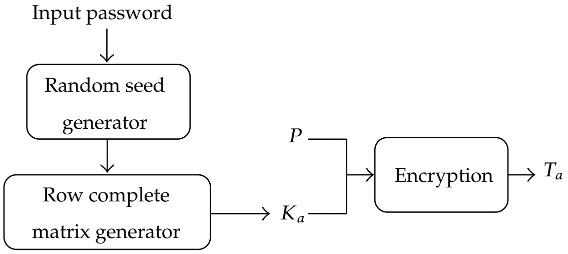

Figure 3: The flowchart of encryption.

Theorem 2.5. *Let A be a row complete matrix derived according to the flowchart in Figure 1. If B is a row complete matrix that is derived from A by columns permutation, then the re-encryption key between A and B can be reduced to the vector form.*

Proof. Since A is derived according to the flowchart 1, A is derived by columns permutation and rows permutation from the simple Latin matrix. That is, if S is the simple Latin matrix, there exists two permutations σ_1 and σ_2 such that $A(i,j) = S(\sigma_1(i),\sigma_2(j))$. Because B is derived from A by column permutations, there exist a permutation σ_3 such that $B(i,j) = A(i,\sigma_3(j))$. Therefore, $B(i,j) = A(i,\sigma_3(j)) = S(\sigma_1(i),\sigma(j))$, where $\sigma = \sigma_2\sigma_1$.

Comparing the formula of A and B, if we want to show that the re-encryption key can be reduced to the vector form, we just need to show that the difference between arbitrary two columns of S is constant. Since $S(i,j) = \text{mod}(i+j,n)$, for given j_1 and j_2, $S(i,j_1) - S(i,j_2) = \text{mod}(j_1 - j_2, n)$ for all $i = 1,\ldots,n$. That is, the difference between the j_1th and j_2th columns of S is a constant $\text{mod}(j_1 - j_2, n)$. Therefore, the re-encryption key between A and B can be reduced to the vector form. □

Corollary 2.6. *Given a row complete matrix A derived according to Figure 1 and a row complete matrix B derived according to Figure 2 with input A, the re-encryption key between A and B can be reduced to vector form.*

In our experiment, if a row complete matrix of size 4 is not generated according to Figure 1, using column permutations can not generate another row complete matrix. However, if the size of row complete matrix is over 6, using column permutations can generate another row complete matrix. It is our on-going project to generate all the row complete matrices so that we can design how many keys are available in this method.

In practice, people often choose a string of characters as a password. It is almost not humanly possible to memorize a Latin matrix as a password, especially when the matrix size is greater than 6-by-6. Hence, in our scheme the password will be mapped to a random seed. Based on this random seed and using the flowchart in Figure 3, the system can generate K_a for Alice.

When Alice wants to share a ciphertext with Bob, Alice uses the flowchart in Figure 4 to generate K_b randomly. At the same time the system can produce the re-encryption key $m_{a,b}$. Before Alice sends K_b to Bob, Alice uses a hashing function to generate a hash code. This hash code, denoted by $h_b = \text{hashing}(K_b)$, plays the role of a password for Bob. Then, Alice will send $m_{a,b}$, the index of the ciphertext and the email address of Bob to the server. At the same time Alice will send K_b and h_b to Bob. When the server receives the re-encryption key and the ciphertext index, it will re-encrypt the ciphertext immediately and send a message to Bob in

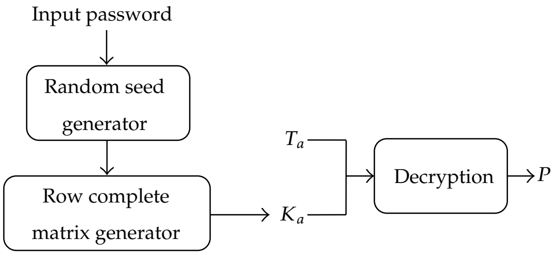

Figure 4: The flowchart of decryption.

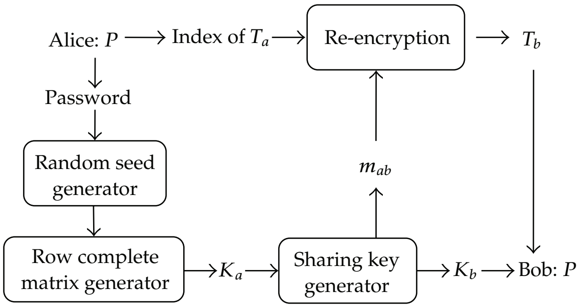

Figure 5: The flowchart of re-encryption.

secure manner, informing him of where to download the ciphertext. When Bob receives the message coming from the server, he can start to download the ciphertext T_b. Bob uses h_b as the password to decrypt ciphertext. The decoding software will use h_b to verify whether K_b matches this hash code. If it matches, the decoding software will start the decryption process of T_b by K_b. That is, the key management is in the client (Bob), not in the cloud (Server). The flowcharts of encryption, re-encryption, and decryption are depicted in Figures 3, 4 and 5, respectively.

3. Security Analysis

In this section we discuss some details in the practical application and the security of our proposed re-encryption method.

To meet the three criteria of re-encryption previously mentioned, the flowchart in Figure 5 only produces a subset of row complete matrices. We have to make sure the number of this subset is sufficiently large such that it is computationally infeasible to guess K_a. When the matrix size is 4, the number of all row complete matrices is 144. When the size is 6, the number rapidly increases to 172800. As the security is strongly related to the size of row complete matrices, we recommend that the matrix size be at least 36. If we only consider the element permutation of the first row and the row permutation of the remaining rows of the row complete matrix, a very conservative estimation of the number of all row complete matrices of size n is $n!(n-1)!$.

When the matrix size is over 32, the conservative estimation of the number of all row complete matrices is about $2.16 * 10^{69}$. Since the number $2.16 * 10^{69}$ is much larger than 2^{128} (around $3.4 * 10^{38}$), we estimate the security strength to be comparable to that of 128-bit block

Table 1: The possible column pairs.

(1)	(2)	(3)
(0, 2)	(3, 0)	(2, 1)
(2, 0)	(0, 1)	(0, 3)
(1, 3)	(2, 3)	(1, 0)
(3, 1)	(1, 2)	(3, 2)

cipher encryption method. The server receives T_a and $m_{a,b}$ for easily producing T_b with a modular addition. However, there is no further information (except for the size of K_a) that can be derived for the server to guess K_a. Even with the knowledge of the tuple $(T_a, T_b, m_{a,b})$, the adversary does not gain more advantage to derive K_a than random guessing. In addition, the probability to guess K_a correctly is at most $1/(n!(n-1)!)$. It is computationally infeasible to derive K_a when $n \geq 32$. This satisfies the first criterion of re-encryption.

For decryption part, Bob receives tuple (T_b, K_b, h_b). The hash code h_b is for Bob to verify K_b so it is highly dependent on K_b, and therefore it is not useful for guessing K_a. K_b is the key for Bob to decrypt plaintext P where $P = D(T_b, K_b)$. Consequently, neither P nor T_b contains the information related to K_a. Hence, the only issue needs to be discussed is whether Bob can use K_b to infer K_a. When the matrix size is n, the length of $m_{a,b}$ is $n-1$ and there are n^{n-1} possibilities for Bob to guess $m_{a,b}$.

If the server does not reveal $m_{a,b}$ to Bob, it is computationally infeasible for Bob to guess the key K_a of Alice when $n \geq 32$. Therefore, our proposed method satisfies the second criterion of the re-encryption.

Finally, in our method, since the server does not store any keys from either the content provider or the receiver, the server does not need to deal with any key management issue. Moreover, the $m_{a,b}$ is just a vector and cost of re-encryption will not increase even when the encryption is strengthened with longer encryption keys, as the complexity of re-encryption is only related to the length of the ciphertext. Hence, we conclude that our proposed method satisfies all three criteria in Section 2.

Although our method fits the requirements for a re-encryption system, we note that if Bob collaborates with the server, they can recover the key of Alice. Specifically, if Bob reveals the plaintext to the server, it can compare the plaintext (P) with T_a and then reconstruct Alice's key K_a. Assume that the keys of Alice and Bob are K_a and K_b, respectively, in (2.2) and the plaintext is $(0,3,2,2,3,0,1,0,1,3,2,0)$. When the server gets this plaintext (P), it compares P with $T_a = (2,0,1,0,0,3,3,1,0,1,3,3)$. From the periodic entries that are encrypted by the first column, the server obtains the mapping pairs $(0,2)$, $(2,0)$, $(1,3)$, and $(3,1)$. Similarly, from the entries that are encrypted by the second column, the server obtains the mapping pairs $(3,0)$, $(0,1)$, and $(2,3)$. Although there are only three pairs, the server can use the nonrepetitiveness property of Latin matrix to find out the fourth pair $(1,2)$. Repeat this process to find out that the pairs of the third column are $(2,1)$, $(0,3)$, $(1,0)$, and $(3,2)$. We can collect the above pairs to form Table 1.

Now, we want to use these three columns of pairs to construct a row complete matrix. Assuming that the column order appearing in the row complete matrix is just (1)-(2)-(3), we have (0, 2), the first column, connecting to (2, 3), the second column, and then (3, 2). Therefore, we get (0, 2, 3, 2). Because the element 2 is repetitive, this row should not be in a latin matrix, and the column order is not (1)-(2)-(3). Hence, we can compare all the possible orders of columns to check whether the latin matrix exists. For this example, we find that there are two possible orders that can form row complete matrices. They are (2)-(1)-(3) and (3)-(1)-(2), and the corresponding row complete matrices are

$$C_1 = \begin{pmatrix} 3 & 0 & 2 & 1 \\ 0 & 1 & 3 & 2 \\ 2 & 3 & 1 & 0 \\ 1 & 2 & 0 & 3 \end{pmatrix}, \qquad C_2 = \begin{pmatrix} 2 & 1 & 3 & 0 \\ 0 & 3 & 1 & 2 \\ 1 & 0 & 2 & 3 \\ 3 & 2 & 0 & 1 \end{pmatrix}. \tag{3.1}$$

These two row complete matrices are not the final results because we have to adjust the orders of their rows. We adjust the candidate matrix C_1 first. When we compare the plaintext and the ciphertext, we observe that the first element 0 is encrypted to 2, and, therefore, we know it is encrypted from the second column to the third column. So the first nonzero element of the first column is 2. The second element of the plaintext is encrypted from 3 to 0, so the second nonzero element of the first column is 1, that is,

$$C_1 = \begin{pmatrix} 2 & 3 & 1 & 0 \\ 1 & 2 & 0 & 3 \\ 3 & 0 & 2 & 1 \\ 0 & 1 & 3 & 2 \end{pmatrix}. \tag{3.2}$$

Because the position of the row leading by 0 has no effect on encryption or decryption, there are four possible row complete matrices for C_1. Similarly, we have

$$C_2 = \begin{pmatrix} 2 & 1 & 3 & 0 \\ 3 & 2 & 0 & 1 \\ 1 & 0 & 2 & 3 \\ 0 & 3 & 1 & 2 \end{pmatrix}. \tag{3.3}$$

In this case, there are four keys that are equivalent in encryption and decryption. How to overcome this problem is an important issue that we will address in the future.

There is another pitfall to watch out for. Ideally the server only re-encrypts the specified ciphertext to Bob by $m_{a,b}$. However, if the server re-encrypts all the ciphertexts of Bob in the cloud by $m_{a,b}$ and allows Bob to access all the re-encrypted files, Bob can see all the plaintexts of Alice.

To resolve this issue, we can modify our encryption method by including an additional parameter that is highly related to the ciphertext, for example, the length of T_a, to change the order of coding. Hence, $m_{a,b}$ is dependent on the ciphertext. The reason we do not modify the encryption/decryption method is to make it easier for the readers to understand this re-encryption method.

4. Conclusions

We proposed a new re-encryption method for the cloud storage service. This method is based on the properties of row complete matrices, which can be beautifully incorporated in designing a practical scheme of cloud storage. It is flexible for the adjustment of the secure requirement; namely, if the level of security needs to be increased, we only have to increase the matrix size. This method is quite suitable for cloud service because the re-encryption cost is rather low and there is no key management issue at all. Hence, we do not have to worry about the rapid growth of contents and there is no need for protecting the shared keys in the server.

There remain some research topics worthy of exploration in the future. For example, our experiments indicate that the flowchart in Figure 1, which generates the row complete matrices will become quite slow when the matrix size is greater than 12. Although the possible permutations can be listed in a look-up table to speed up the matrix generation, to achieve significant improvements it is probably preferred to design faster algorithms for generating the row complete matrices. The other problem is the collaboration between Bob and Server. In the example of security analysis we can see that some row complete matrices are equivalent in the sense of encryption and decryption, and we have to compute carefully all the possible numbers of equivalent row complete matrices to make sure of the security level. However, this property can be used in the collaborative writing, in which users with different keys can edit the same document at the same time [15].

References

[1] C. J. Colbourn, T. Kløve, and A. C. H. Ling, “Permutation arrays for powerline communication and mutually orthogonal Latin squares,” *IEEE Transactions on Information Theory*, vol. 50, no. 6, pp. 1289–1291, 2004.

[2] A. Caprara, “Sorting by reversals is difficult,” in *Proceedings of the 1st Annual International Conference on Computational Molecular Biology (RECOMB ’97)*, pp. 75–83, ACM, January 1997.

[3] J. Kong, “The role of latin square in cipher systems: a matrix approach to model encryption modes of operation,” UCLA Computer Science Department Technical Report number 030038, 2008.

[4] M. Blaze, G. Bleumer, and M. Strauss, “Divertible protocols and atomic proxy cryptography,” in *EuroCrypt 1998*, vol. 1403 of *Lecture Notes in Computer Science*, pp. 127–144, Springer, Heidelberg, Germany, 1998.

[5] J. Weng, R. H. Deng, X. Ding, C. K. Chu, and J. Lai, “Conditional proxy re-encryption secure against chosen-ciphertext attack,” in *Proceedings of the 4th International Symposium on ACM Symposium on Information, Computer and Communications Security (ASIACCS ’09)*, pp. 322–332, March 2009.

[6] S. S. Vivek, S. Sharmila Deva Selvi, V. Radhakishan, and C. Pandu Rangan, “Conditional proxy re-encryption—a more efficient construction,” *Communications in Computer and Information Science*, vol. 196, pp. 502–512, 2011.

[7] B. Libert and D. Vergnaud, “Unidirectional chosen-ciphertext secure proxy re-encryption,” *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1786–1802, 2011.

[8] X. A. Wang, X. Huang, X. Yang, L. Liu, and X. Wu, “Further observation on proxy re-encryption with keyword search,” *Journal of Systems and Software*, vol. 85, no. 3, pp. 643–654, 2012.

[9] M. Green and G. Ateniese, “Identity-based proxy re-encryption,” in *Proceedings of the 5th International Conference on Applied Cryptography and Network Security*, vol. 4521 of *Lecture Notes in Computer Science*, pp. 288–306, 2007.

[10] C. K. Chu and W. Tzeng, “Identity-based proxy re-encryption without randomoracles,” in *Proceedings of the Information Security Conference*, vol. 4779 of *Lecture Notes in Computer Science*, pp. 189–202, 2007.

[11] Q. Liu, C. C. Tan, J. Wu, and G. Wang, “Reliable re-encryption in unreliable clouds,” in *Proceedings of the IEEE Global Communications Conference*, Houston, Tex, USA, December 2011.

[12] L. Fang, W. Susilo, C. Ge, and J. Wang, “Interactive conditional proxy re-encryption with fine grain policy,” *Journal of Systems and Software*, vol. 84, no. 12, pp. 2293–2302, 2011.

[13] G. Ateniese, K. Benson, and S. Hohenberger, “Key-private proxy re-encryption,” in *Topics in Cryptology—CT-RSA 2009*, vol. 5473 of *Lecture Notes in Computer Science*, pp. 279–294, Springer, Berlin, Germany, 2009.

[14] J. Shao, P. Liu, and Y. Zhou, “Achieving key privacy without losing CCA security in proxy re-encryption,” *Journal of Systems and Software*, vol. 85, no. 3, pp. 655–665, 2012.

[15] R. A. Calvo, J. Jones, K. Yacef, and P. Reimann, “Collaborative writing support tools on the cloud,” *IEEE Transactions on Learning Technologies*, vol. 4, no. 1, pp. 88–97, 2011.