參數式列完備矩陣之轉加密方法

蔡哲民[1]、曾正男[2]、陳以德[3*]
[1]崑山科技大學資訊傳播系
[2]國立政治大學應用數學系
[3*]高雄醫學大學醫務管理暨醫療資訊系
[1]tjm@mail.ksu.edu.tw、[2]jengnan@gmail.com、[3*]itchen@kmu.edu.tw

摘要

拉丁矩陣是密碼學領域中，一個常用的數學型式，而列完備矩陣(Row-complete Matrix)是在 Z_n 的拉丁矩陣的特殊形式。我們發現列完備矩陣，可以應用在雲端傳輸資料的轉加密(Re-encryption)系統中。本文中，我們探討更多列完備矩陣的屬性；並提出一個參數式列完備矩陣的轉加密方法，來加強雲端再加密系統的安全性。

關鍵詞：雲端轉加密、密碼方法、列完備矩陣

Re-Encryption Method based on parameter Row-complete Matrix

Jer-Min Tsai[1], Jengnan Tzeng[2], and I-Te Chen[3*]
[1]Kun Shan University, Tainan 71003, TAIWAN
[2]National Cheng-Chi University, Taipei 11605, TAIWAN
[3]Kaohsiung Medical University, Kaohsiung 80708, TAIWAN
E-mail: tjm@mail.ksu.edu.tw[1], jengnan@gmail.com[2], itchen@kmu.edu.tw[3*]

Abstract

Latin matrix is one of mathematical type in the field of cryptography; in addition, Row-complete Matrix is a special form of Z_n Latin matrix. The Row-complete Matrix can be used in clouds re-encryption system. In this paper, we discuss the properties of Row-complete Matrix; and propose a re-encryption method based on parameter Row-complete Matrix to strengthen the security of clouds re-encryption system.

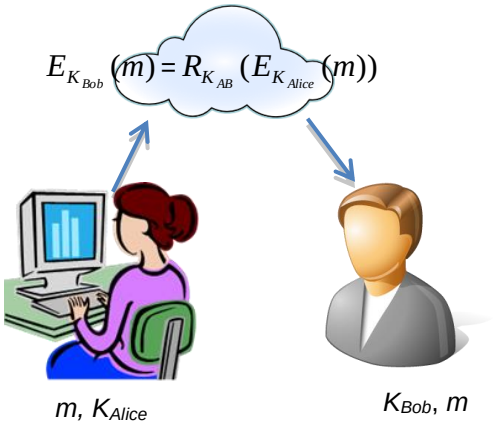
Keywords: Cloud Re-encryption, Cryptography, Row-complete matrix

[*]通訊作者 (Corresponding author.)

壹、 前言

雲端是近來快速發展的應用，透過雲端，使用者可以同步手機、平板、桌上型電腦等系統；且只要有網路，就可以下載雲端資料到任何電腦、手機或平板繼續工作。而影響人們使用雲端系統意願中，最重要的就是它的安全性問題。我們或許可以透過以往發展的對稱及非對稱加密機制，來達到雲端加、解密的目的；但若要將雲端資料分享給他人，用傳統的加密機制，就得先下載、解密、用對方的金鑰加密、上傳到雲端、再通知對方下載、解密。不但費時、費工，且浪費網路頻寬。

舉例來說，Alice 用自己的金鑰 K_{Alice}，及加密方法 E，加密一份明文 m，成為 $E_{K_{Alice}}(m)$，上傳到雲端。若 Alice 要將 m 傳給 Bob，以傳統方式，Alice 要先將 $E_{K_{Alice}}(m)$ 下載回來，解密回 m，再用 Bob 的金鑰 K_{Bob} 或 Alice 及 Bob 事先商量好的 Session Key K_{SAB} 加密成為 $E_{K_{Bob}}(m)$ 或 $E_{K_{SAB}}(m)$ 後，上傳到雲端，再由 Bob 下載 $E_{K_{Bob}}(m)$ 或 $E_{K_{SAB}}(m)$ 回來解密。此傳統方式，不但浪費 CPU 資源且浪費網路頻寬。於是 Proxy Re-encryption(PRE)轉加密機制就因應而生。PRE 機制期望使用者(如 Alice)在雲端上的密文，能經過轉加密的機制，由雲端伺服器，轉加密成他人(如 Bob)能接收，且可以解密的密文，如圖一所示。



圖一： 轉加密機制

以上述例子而言，若是雲端伺服器，可以將 Alice 上傳到雲端的密文 $E_{K_{Alice}}(m)$，直接經由轉加密演算法 R 及 Alice 給雲端伺服器的轉加密金鑰 (re-encryption key)K_{AB}，轉換成 Bob 可以解密的 $E_{K_{Bob}}(m)$；亦即：

$$E_{K_{Bob}}(m) = R_{K_{AB}}(E_{K_{Alice}}(m)) \quad \ldots\ldots (1)$$

如此一來，Alice 就不用下載密文、解密回 m、加密成 Bob 可解密的密文格式再上傳；省去 2 段上、下載及 Alice 解、加密的運算時間；雲端伺服器也只做一次的轉加密運算而已；Bob 則跟傳統方法一樣，下載密文回來解密即可。

貳、文獻探討

Blaze 等學者在 1998 年提出基於 Elgamal 的 Proxy Re-encryption 轉加密機制 [2]，此機制可把 Alice 加密過的密文，用給定的 Re-encryption key，轉加密為 Bob 可以解開的密文。在傳統的轉加密機制中，雲端可以將「任何」Alice 的密文，轉加密成 Bob 可解讀的密文；這樣不肖的雲端，可以把所有 Alice 在雲端上的密文，統統轉加密給 Bob。

因此，Weng 等學者[10]，在 2009 年提出了一個條件式的轉加密機制(Conditional PRE)，雲端只能在某些條件下，轉加密 Alice 在雲端上的密文給 Bob。不幸的是，Weng 等學者的條件式轉加密機制後來被破解了。Vivek 等學者在 2011 年對條件式轉加密機制做了改進，變得有效率些，且做了一些安全性的證明[8]。

為了確認身分，通常會由公正第三方產生轉加密金鑰，或是使用憑證的機制；2007 年 Green 及 Ateniese 學者提出了 Identity-based 的轉加密機制[5]，以使用者的 ID 來做身份確認。在 2011 年，Wan 等五位學者對 Identity-based 及 Mediated Identity -Based 做了個比較與探討[9]；2015 年 Akshayaram Srinivasan 及 C. Pandu Rangan，提出了一個 Certificateless PRE 機制，不用憑證也可以達到認證的目的[1]。另一方面，學者 Liu 於 2014 年提出基於時間(Time-based)的轉加密機制[7]；此機制把屬性加密(attribute-based)與 PRE 結合，雲端伺服器指定一段時間間隔，雲端伺服器會檢查 Bob 訪問屬性和相應屬性的有效時間，過了有效時間，Bob 下載的任何檔案，便不能再解密回來了。2015 年，T. Mariya John 學者，提出了一個 Threshold Proxy Re-Encryption Scheme，透過分散式 erasure code，來達成一個安全的分散式雲端儲存系統[6]。

在 2012 年，Chen 等三位學者，提出了 Re- Encryption Method Designed by Row Complete Matrix[3]，這是首篇使用列完備矩陣(Row Complete Matrix) 來建構 PRE 機制的期刊論文，在文中亦提到若 Bob 跟雲端伺服器共謀，可以解出 Alice 的金鑰；我們亦發現，在 Chen 的 PRE 機制中，不肖的雲端，可以把所有 Alice 在雲端上的密文，統統轉加密給 Bob。為了解決這兩個問題，本文將先介紹在 2012 年，由 Chen 等三位學者所提出的 PRE 機制，進而對此機制提出改良方法，解決上述兩個問題，提升雲端 PRE 機制的安全性。

參、先備知識

拉丁矩陣是在 Z_n 底下的 $n*n$ 方陣，在每一行和每一列，該元素只會出現一次，正交拉丁矩陣可以用在糾錯碼和數學難題如數獨上。Colbourn 在 2004 年證明，部分填充數獨問題，是一個 NP- complete 的問題 [4]。此外，拉丁矩陣 multi-permutation 方法，在密碼學上有很重要的應用。

3.1 列完備矩陣

而列完備矩陣是拉丁矩陣的一個特例，首先是由內華達大學數學系 Dr. Peter Shiue 所定義。列完備矩陣，對任一(*i, j*)-pair 屬於 Z_n，其中 $i \neq j$，這對(*i, j*)-pair，只會存在於拉丁矩陣的某一列正好一次；但若以行來看，則(*i, j*)-pair，有可能會存在 1 次以上。同理，若為行完備矩陣，則(*i, j*)-pair，只會存在於拉丁矩陣的某一行恰好一次。以圖二舉例來說，如下面這個矩陣，在每一行和每一列，該元素只會出現一次，所以是個拉丁矩陣；而且對(0,1)、(1,2)及(2,3)-pair 屬於 Z_4，分別在第 1、4 及 3 列出現，且在整個矩陣的列中，只出現一次。但若以行來看，(0,3)-pair，則在第 2、3、4 行都出現，因此，此矩陣不為行完備矩陣。

$$\begin{bmatrix} 3 & 2 & 0 & 1 \\ 2 & 1 & 3 & 0 \\ 1 & 0 & 2 & 3 \\ 0 & 3 & 1 & 2 \end{bmatrix}$$

圖二：列完備矩陣(Row Complete Matrix)

當 *n*=2 時，僅有 2 個列完備矩陣；當 *n*=4 時，則有 144 個列完備矩陣；當 *n*=6 時，列完備矩陣增加到 172800 個。Dr. Peter Shiue 除了給列完備矩陣的定義外，也提出了列完備矩陣的特性。當時他觀察到，列完備矩陣只存在於矩陣大小為偶數的矩陣。若矩陣大小為奇數如 *n*=3 或 *n* =5，則不存在列完備矩陣。因此，他提出了一個開放的問題，除了 2 以外，當矩陣大小為質數時，是否就不存在列完備矩陣。我們將會對這個開放的問題，提出我們的看法，並試著證明列完備矩陣只存在於矩陣大小是偶數的矩陣。

3.2 列完備矩陣的 PRE 機制

2012 年 Chen 等三位學者所提出的 PRE 機制[3]，Alice 自己的金鑰為 K_a，當 Alice 想分享她的內容（P）給 Bob，她將創建兩個 Key：一把 K_b 是 Bob 可以從轉加密後的密文解密回來的 Key；一把 $m_{a,b}$，它只是一個向量，用來請雲端轉加密密文（T_a）成為 Bob 可以解開（T_b）的 Key。Chen 等定義 E 是加密，D 是解密，R 是轉加密，則有下列的關係：

$$T_a = \mathrm{E}(\mathrm{P}, K_a),$$

$$\mathrm{P} = \mathrm{D}(T_a, K_a),$$

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

$$\mathrm{P} = \mathrm{D}(T_b, K_b).$$

K_a 為一 *n***n* 的列完備矩陣，$K_a(s, t)$即 K_a 中的 *s* row 跟 *t* column 的元素；其中 $t = \mathrm{mod}(k-1, n-1) + 1$，*s* 為明文在 *t* column 中的索引，mod 為同餘運算(congruence modulo)，亦

即 $K_a(s, t)$ = P(k)。加密時，則 n-base 的明文第 k 個元素，P(k)將被加密為 $K_a(s,t+1)$。舉例來說，有個列完備矩陣 K_a 是 4-base 的如下所示：

$$K_a = \begin{bmatrix} 3 & 2 & 0 & 1 \\ 2 & 1 & 3 & 0 \\ 1 & 0 & 2 & 3 \\ 0 & 3 & 1 & 2 \end{bmatrix} \quad \text{..... (2)}$$

加密元素分別為 3、0、2，第一 column 是決定用那一個 column 的索引，在 K_a 中，第一 column 為 3、2、1、0，所以加密順序，分別由(3,4)(2,3)及(1,2) column-pair 決定。第一個元素 3，將由 K_a (3,4) column-pair 決定，3 在第 3 column 中的第 1 個位置，3 將被加密成第 4 column 中同一 row 的 0；第 2 個元素 0 由 K_a 的(2,3)column-pair 決定，0 在第 2 column 中的第三個位置，0 將被加密成第 3 column 中的 2，第 3 個元素 2 由 K_a 的 (1,2) column-pair 決定，2 在第 1 column 中的第 1 個位置，2 將被加密成第 2 column 中的 1。第 4 個元素又回到第 1 及 2 column 所決定，如此循還加密下去；解密時，則反向操作即可。當然當矩陣大小為 4 的列完備矩陣，是不夠安全的，Chen 等建議，至少使用矩陣大小為 36 以上的列完備矩陣，以提高安全性。

轉加密時，若 K_a 同樣為式(2)、K_b 如式(3)所示，

$$K_b = \begin{bmatrix} 1 & 0 & 2 & 3 \\ 2 & 1 & 3 & 0 \\ 3 & 2 & 0 & 1 \\ 0 & 3 & 1 & 2 \end{bmatrix} \quad \text{..... (3)}$$

Chen 等找出以 K_a、K_b 在加解密時，$m_{a,b}$ 如下所示：

$$m_{a,b} = \begin{bmatrix} 1 & 1 & 1 & 2 \\ 2 & 1 & 1 & 2 \\ 3 & 1 & 1 & 2 \\ 0 & 1 & 1 & 2 \end{bmatrix} \quad \text{..... (4)}$$

$m_{a,b}$ 差了一個向量(1,1,2)，也就是在 Base-4 底下，用 K_a 加密的密文，每個值重覆加向量(1,1,2)，就可以讓 Bob 把密文解回來。所以，只要把向量(1,1,2)送給雲端，雲端只要對 T_a 中的元素重覆做加(1,1,2)的動作，則可以把 T_a 轉成 Bob 可以用 K_b 解開的 T_b。Bob 無法從 K_b 直接計算出 K_a，雲端只有 $m_{a,b}$ 也無法算出 K_a、K_b，並且整個加解密運算都只用到加、減、取餘數等簡單的演算法，以致效能良好，適合加密大量資料，尤其是影音的資料。但若 Bob 與雲端共謀，就可以解出 K_a。

肆、參數式列完備矩陣 PRE 機制

Chen 等的列完備矩陣 PRE 機制，若 Bob 跟雲端伺服器共謀，可以解出 Alice 的金鑰；我們亦發現，在 Chen 的 PRE 機制中，不肖的雲端，可以把所有 Alice 在雲端上的密文，統統轉加密給 Bob。針對這 2 個問題，本文提出了改良之參數式列完備矩陣 PRE 機制。

4.1 系統參數

本文使用的參數符號，與原 Chen 等學者 2012 年提出的轉加密機制一致，但使用方式有所修正，參數符號如表一所示：

表一: 參數式列完備矩陣 PRE 機制參數符號

P	明文
T_a'	用 K_s Rotate 後的暫時明文
T_a	Alice 用 K_a 加密後的密文
T_b	雲端轉加密後要給 Bob 解密的密文
K_a	Alcie 的列完備矩陣私鑰，
K_b	Alcie 原讓 Bob 解密的列完備矩陣私鑰
$m_{a,b}$	給雲端轉加密用之金鑰
K_s	對明文及 K_b 做 permutation 之參數，為本文最重要之參數
K_d	K_b 經由 K_s 的重排後所產生實際讓 Bob 解密之金鑰

E 是加密，D 是解密，R 是轉加密，Rotate 是擾亂明文的方式，則我們定義下列式子：

$$T_a' = \text{Rotate}(P, K_s),$$

$$T_a = E(T_a', K_a),$$

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

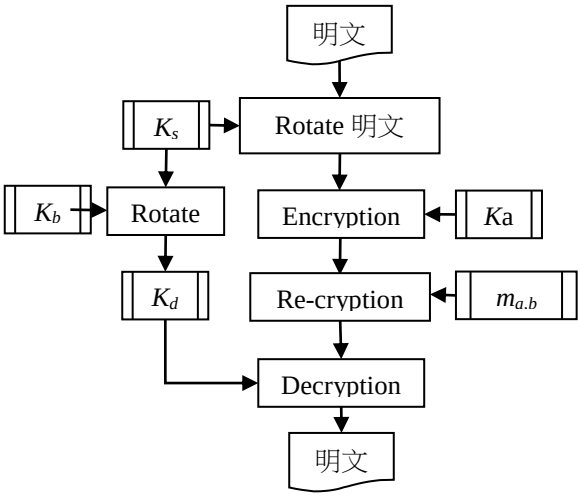
$$K_d = \text{Rotate}(K_b, K_s)$$

$$P = D(T_b, K_d)$$

4.2 參數式列完備矩陣 PRE 機制

K_a、K_b 為亂數產生的列完備矩陣，$m_{a,b}$ 為 K_a、K_b 的差值，要給雲端轉加密用。我們在原 Chen 等學者 2012 年提出的轉加密機制中，加入一組參數 key K_s，用來擾亂明文及 K_b。明文在加密前，要分別先加入 K_s 的值，再由 K_a 加密成新的密文；假設矩陣大小

為 dim，產生 K_d 的方法，是由 K_b 每一個元素 x，由矩陣大小減 1(即為 dim-1)開始往前做處理，在 mod dim 的運算下，減掉 K_s 由 0 往 dim-1 的值。演算法如下圖所示：



圖三：列完備矩陣 PRE 機制演算法

Alice 製造出 K_a、K_b、$m_{a,b}$、K_s 及 K_d，當 Alice 要將一份用 K_a 加密過放在雲端的密文傳給 Bob 時，Alice 將 $m_{a,b}$ 傳給雲端，並將 K_d 傳給 Bob。
加密後的密文，Bob 需拿 K_d 來解密，才能解回原來的明文。

4.3 參數式列完備矩陣 PRE 機制 4.3 實例說明

我們令參數 key K_s: 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4

$$K_a = \begin{bmatrix} 0 & 6 & 1 & 13 & 5 & 7 & 10 & 4 & 11 & 15 & 14 & 12 & 9 & 2 & 3 & 8 \\ 1 & 7 & 2 & 14 & 6 & 8 & 11 & 5 & 12 & 0 & 15 & 13 & 10 & 3 & 4 & 9 \\ 2 & 8 & 3 & 15 & 7 & 9 & 12 & 6 & 13 & 1 & 0 & 14 & 11 & 4 & 5 & 10 \\ 3 & 9 & 4 & 0 & 8 & 10 & 13 & 7 & 14 & 2 & 1 & 15 & 12 & 5 & 6 & 11 \\ 4 & 10 & 5 & 1 & 9 & 11 & 14 & 8 & 15 & 3 & 2 & 0 & 13 & 6 & 7 & 12 \\ 5 & 11 & 6 & 2 & 10 & 12 & 15 & 9 & 0 & 4 & 3 & 1 & 14 & 7 & 8 & 13 \\ 6 & 12 & 7 & 3 & 11 & 13 & 0 & 10 & 1 & 5 & 4 & 2 & 15 & 8 & 9 & 14 \\ 7 & 13 & 8 & 4 & 12 & 14 & 1 & 11 & 2 & 6 & 5 & 3 & 0 & 9 & 11 & 15 \\ 8 & 14 & 9 & 5 & 13 & 15 & 2 & 12 & 3 & 7 & 6 & 4 & 1 & 10 & 11 & 0 \\ 9 & 15 & 10 & 6 & 14 & 0 & 3 & 13 & 4 & 8 & 7 & 5 & 2 & 11 & 12 & 1 \\ 10 & 0 & 11 & 7 & 15 & 1 & 4 & 14 & 5 & 9 & 8 & 6 & 3 & 12 & 13 & 2 \\ 11 & 1 & 12 & 8 & 0 & 2 & 5 & 15 & 6 & 10 & 9 & 7 & 4 & 13 & 14 & 3 \\ 12 & 2 & 13 & 9 & 1 & 3 & 6 & 0 & 7 & 11 & 10 & 8 & 5 & 14 & 15 & 4 \\ 13 & 3 & 14 & 10 & 2 & 4 & 7 & 1 & 8 & 12 & 11 & 9 & 6 & 15 & 0 & 5 \\ 14 & 4 & 15 & 11 & 3 & 5 & 8 & 2 & 9 & 13 & 12 & 10 & 7 & 0 & 1 & 6 \\ 15 & 5 & 0 & 12 & 4 & 6 & 9 & 3 & 10 & 14 & 13 & 11 & 8 & 1 & 2 & 7 \end{bmatrix} \quad \text{.....(5)}$$

$$K_b = \begin{bmatrix} 14 & 4 & 15 & 11 & 3 & 5 & 8 & 2 & 9 & 13 & 12 & 10 & 7 & 0 & 1 & 6 \\ 1 & 7 & 2 & 14 & 6 & 8 & 11 & 5 & 12 & 0 & 15 & 13 & 10 & 3 & 4 & 9 \\ 5 & 11 & 6 & 2 & 10 & 12 & 15 & 9 & 0 & 4 & 3 & 1 & 14 & 7 & 8 & 13 \\ 3 & 9 & 4 & 0 & 8 & 10 & 13 & 7 & 14 & 2 & 1 & 15 & 12 & 5 & 6 & 11 \\ 4 & 10 & 5 & 1 & 9 & 11 & 14 & 8 & 15 & 3 & 2 & 0 & 13 & 6 & 7 & 12 \\ 0 & 6 & 1 & 13 & 5 & 7 & 10 & 4 & 11 & 15 & 14 & 12 & 9 & 2 & 3 & 8 \\ 13 & 3 & 14 & 10 & 2 & 4 & 7 & 1 & 8 & 12 & 11 & 9 & 6 & 15 & 0 & 5 \\ 10 & 0 & 11 & 7 & 15 & 1 & 4 & 14 & 5 & 9 & 8 & 6 & 3 & 12 & 13 & 2 \\ 9 & 15 & 10 & 6 & 14 & 0 & 3 & 13 & 4 & 8 & 7 & 5 & 2 & 11 & 12 & 1 \\ 8 & 14 & 9 & 5 & 13 & 15 & 2 & 12 & 3 & 7 & 6 & 4 & 1 & 10 & 11 & 0 \\ 2 & 8 & 3 & 15 & 7 & 9 & 12 & 6 & 13 & 1 & 0 & 14 & 11 & 4 & 5 & 10 \\ 11 & 1 & 12 & 8 & 0 & 2 & 5 & 15 & 6 & 10 & 9 & 7 & 4 & 13 & 14 & 3 \\ 12 & 2 & 13 & 9 & 1 & 3 & 6 & 0 & 7 & 11 & 10 & 8 & 5 & 14 & 15 & 4 \\ 7 & 13 & 8 & 4 & 12 & 14 & 1 & 11 & 2 & 6 & 5 & 3 & 0 & 9 & 10 & 15 \\ 6 & 12 & 7 & 3 & 11 & 13 & 0 & 10 & 1 & 5 & 4 & 2 & 15 & 8 & 9 & 14 \\ 15 & 5 & 0 & 12 & 4 & 6 & 9 & 3 & 10 & 14 & 13 & 11 & 8 & 1 & 2 & 7 \end{bmatrix} \text{.....(6)}$$

由 K_a 及 K_b，我們計算 K_a 及 K_b 加、解密時的差值，我們可以算出 $m_{a,b}$ 如下式(7)所示。舉例來說，K_b 第一個數字為"14"，K_a 中，第一行的"14"會換成"4"，K_b 解密是從最後一行往回看，而 K_b 的最後第 2 行"14"的，是由最後一行"3"變回來的，但原來 K_a 是從"14"變成"4"，所以 $m_{a,b}$ 第一個參數就是
(3-4) mod 16 = 15；再則，K_b 第二行第一個數字為"1"，K_a 中，第 2 行的"1"會換成"12"，K_b 解密第二個字是從最後第二行往回看，而 K_b 的最後第 3 行"1"的，是由最後第 2 行"2"變回來的，但原來 K_a 是從 1 變成 12，所以 $m_{a,b}$ 第二個參數就是(2-12) mod 16 = 6，以此類推，詳細演算法，請參閱[7]。

$$m_{a,b} = 15\ 6\ 13\ 5\ 12\ 12\ 10\ 0\ 6\ 4\ 4\ 11\ 3\ 10\ 1\ 0\ldots\ldots \quad (7)$$

對於矩陣大小為 16 方陣而言，產生 K_d 的方法是對 K_b 每一個元素 x，由矩陣大小減 1 的那 一行開始往前處理 (此例為 16-1=15，由第 15 行亦即倒數第二行開始處理)；在 mod 16 的運算下，將整行依序減掉 K_s 由 0 往 16-1 的值而得到。以此例而言，K_s 為 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4，我們從第 15 行開始處理，第 15 行全部減掉 1 (K_s 的第一個值為 1)；第 14 行全部減掉 1+2=3 (K_s 的第一個及第二個值分別為 1、2，加起來等於 3)；第 13 行全部減掉 1+2+3=6，第 12 行全部減掉 1+2+3+4=10；以此類推來得到 K_d。

$$K_d = \begin{bmatrix} 0 & 9 & 6 & 3 & 15 & 4 & 9 & 4 & 15 & 6 & 7 & 6 & 7 & 3 & 6 & 12 \\ 1 & 10 & 7 & 4 & 0 & 5 & 10 & 5 & 0 & 7 & 8 & 7 & 8 & 4 & 7 & 13 \\ 2 & 11 & 8 & 5 & 1 & 6 & 11 & 6 & 1 & 8 & 9 & 8 & 9 & 5 & 8 & 14 \\ 3 & 12 & 9 & 6 & 2 & 7 & 12 & 7 & 2 & 9 & 10 & 9 & 10 & 6 & 9 & 15 \\ 4 & 13 & 10 & 7 & 3 & 8 & 13 & 8 & 3 & 10 & 11 & 10 & 11 & 7 & 10 & 0 \\ 5 & 14 & 11 & 8 & 4 & 9 & 14 & 9 & 4 & 11 & 12 & 11 & 12 & 8 & 11 & 1 \\ 6 & 15 & 12 & 9 & 5 & 10 & 15 & 10 & 5 & 12 & 13 & 12 & 13 & 9 & 12 & 2 \\ 7 & 0 & 13 & 10 & 6 & 11 & 0 & 11 & 6 & 13 & 14 & 13 & 14 & 10 & 13 & 3 \\ 8 & 1 & 14 & 11 & 7 & 12 & 1 & 12 & 7 & 14 & 15 & 14 & 15 & 11 & 14 & 4 \\ 9 & 2 & 15 & 12 & 8 & 13 & 2 & 13 & 8 & 15 & 0 & 15 & 0 & 12 & 15 & 5 \\ 10 & 3 & 0 & 13 & 9 & 14 & 3 & 14 & 9 & 0 & 1 & 0 & 1 & 13 & 0 & 6 \\ 11 & 4 & 1 & 14 & 10 & 15 & 4 & 15 & 10 & 1 & 2 & 1 & 2 & 14 & 1 & 7 \\ 12 & 5 & 2 & 15 & 11 & 0 & 5 & 0 & 11 & 2 & 3 & 2 & 3 & 15 & 2 & 8 \\ 13 & 6 & 3 & 0 & 12 & 1 & 6 & 1 & 12 & 3 & 4 & 3 & 4 & 0 & 3 & 9 \\ 14 & 7 & 4 & 1 & 13 & 2 & 7 & 2 & 13 & 4 & 5 & 4 & 5 & 1 & 4 & 10 \\ 15 & 8 & 5 & 2 & 14 & 3 & 8 & 3 & 14 & 5 & 6 & 5 & 6 & 2 & 5 & 11 \end{bmatrix} \text{..... (8)}$$

以明文 P = "4、3、2、1"為例，加密時，第一個明文"4"先加 K_s 中的 1 成為"5"，再 K_a 的第(1,2) column-pair，column 1 中的"5"，會對到 çolumn 2 中的"11"，密文便為"11"；第二個明文"3"先加 K_s 中的 2 成為"5"，再 K_a 的第(2,3) column-pair，column 2 中的"5"，

會對到 column 3 中的"0"，密文便為"0"；同理，第三、四個明文"21"將分別轉為密文"1"及"13"，也就是 T_a = "11、0、1、13"。

轉加密時，$m_{a,b}$ 的做法跟 Chen 等 2012 提出的方法相同，"11、0、1、13"經雲端伺服器轉換密文分別加上 $m_{a,b}$ 第二行的 "15、6、13、5" 變成"10、6、14、2"，詳細過程如下：

11+15 mod 16 = 26 mod 16=10
0+ 6 mod 16 = 6
1+13 mod 16 = 14
13+ 5 mod 16 = 18 mod 16 = 2
也就是 T_b = "10、6、14、2"

Bob 要解密時，T_b "10、6、14、2" 由 K_d 尾端的 Column 開始，第一個密文由 column (16,15) pair 決定，在 column 16 中，"10" 那個位置的 column 15 是"4"；第二個密文由 column (15,14) pair 決定，在 column 15 中，"6" 那個位置的 column 14 是"3"；同理，第三、四個密文分別會轉回"2、1"，如此便把明文"1、2、3、4"解密回來了.。

伍、結論

當矩陣大小為 n 時，約有 $n!(n-1)!$ 個列完備矩陣，當矩陣大小為 32 時，約有 $2.14*10^{69}$ 個列完備矩陣，這比 2^{128} 約等於 $3.4*10^{38}$ 還大，也就是比 128-bit block cipher 的對稱式加密方法的金鑰數量還多，所以建議使用矩陣大小大於 32 以上的列完備矩陣比較安全；$n*n$ 的列完備矩陣到底有幾個，仍是個開放式問題，有興趣的學者，可以再做更進一步的探討。

2012 年，Chen 等的論文中提到若 Bob 跟雲端伺服器共謀，可以解出 Alice 的金鑰；除此之外，我們亦發現，在 Chen 的 PRE 機制中，不肖的雲端，可以把所有 Alice 在雲端上的密文，統統轉加密給 Bob。本文改良加入 K_s 這個參數，將明文先 rotate 一次，並對 K_b rotate 成為 K_d，此舉雖多保管一把長度為 n 的向量 K_s，但 Bob 拿到的 K_d 無法與 Alice 給雲端伺服器的 $m_{a,b}$ 共謀而算出 K_a；破解金鑰的難度也要多* $n!$倍，也就是$(n!)^2(n-1)!$。

同時，Alice 給雲端伺服器的 $m_{a,b}$ 跟 2012Chen 等提出的 PRE 機制是一樣的，但要給 Bob 的密文，是經過 K_s Rotate 後才去雲端伺服器加密，Alice 其他的密文，仍是只用 K_a 加密。因此，就算雲端伺服器把 Alice 所有的密文，全部用 $m_{a,b}$ 轉加密成 T_b，Bob 也只能用 K_d 解密出有經過 K_s Rotate 過的密文，其他的密文解回來是一堆亂碼。

本論文解決了 Bob 跟雲端伺服器共謀與不肖雲端，把所有 Alice 在雲端上的密文，統統轉加密給 Bob 的兩個問題。加入 K_s、K_d 等參數運算，只比 2012 年 Chen 等學者提出的多了幾個同餘的加法運算及 rotate，卻解決其 2 個重要的安全性問題，為本文的主要貢獻。

參考文獻

[1] Akshayaram, Srinivasan and C. Pandu Rangan, "Certificateless Proxy Re-Encryption without Pairing: Revisited," *Proceedings of the 3rd International Workshop on Security in Cloud Computing*, pp. 41-52, April 14th 2015.

[2] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography." *Eurocrypt '98*, pp. 127–144, 1998.

[3] I-Te Chen, Jer-Min Tsai and Jengnan Tzeng," Re-Encryption Method Designed by Row Complete Matrix." *Mathematical Problems in Engineering*, Vol. 2012, Article ID 402890, April 2012.

[4] 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.

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

[6] T. M. John, "Concealing Data Files in the Cloud (Threshold Proxy Re-Encryption Scheme)," *International Journal of Emerging Technology and Innovative Engineering*, Volume I, Issue 3, pp. 59-62, March 2015.

[7] Qin Liu, Guojun Wang, Jie Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Information Sciences*, Vol. 258, pp. 355–370, Feb. 2014.

[8] 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.

[9] Zhong-Mei Wan, Jian Weng, Xue-Jia Lai, Sheng-Li Liu AND Ji-Guo Li, "On the Relation between Identity- Based Proxy Re-Encryption and Mediated Identity- Based Encryption," *Journal of Information Science and Engineering*, Vol.27, pp.243-259, 2011.

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