應用於雲端運算之可驗證門檻式雲端儲存系統

李南逸
南臺科技大學資訊工程系
台南市永康區南台街 1 號
nylee@stust.edu.tw

張宗典
南臺科技大學資訊管理所
台南市永康區南台街 1 號
ma090108@stust.edu.tw

摘要

目前雲端運算之應用已經隨處可見，如我們常用的桌上型電腦、筆記型電腦、智慧型手機甚至平板電腦等行動裝置，都可以透過網路連到雲端伺服器做資料的存取、更新或複雜的計算，使用者的終端設備變成一種資訊的輸入與輸出的媒介，任何人都可以透過自己的終端設備享受到高效能的軟/硬體服務，但也因為將資料置於雲端儲存伺服器，使得資料安全的問題逐漸顯露出來，如何保護使用者放置在雲端伺服器的資料已變成一個重要的課題。本論文擬發展並設計新的雲端伺服器儲存機制，主要是利用混合式密碼系統，讓使用者下載/解密資料時能更加有效率，此外，也提出一套驗證機制，讓使用者能識別雲端伺服器是否為欺騙者或有共謀的問題。

關鍵詞：雲端運算、資料安全、混合式密碼系統、驗證機制、門檻式系統

一、 簡介

近年來，由於電腦與網路的快速發展，使用者的資料越來越多，使用者之間的資源需要互相分享等等，造成資料儲存空間之不足。因此雲端儲存服務供應商為此開始提供相關服務。新興的模式使得使用者可利用雲端環境做資料運算與資料上傳/下載，享受它所提供的服務，而使用者並不需要知道有多少雲端伺服器提供這個服務，因此愈來愈多的使用者傾向將資料儲存於雲端伺服器中，但也因將資料儲存於雲端伺服器中，使得資料安全的問題逐漸顯露出來，如何保護使用者儲存於雲端伺服器的資料已變成一個重要的課題。

欲保護機密文件內容不洩漏出去，最直接的方法就是將機密文件加密，再上傳至雲端伺服器中，接下來就是如何保護金鑰，避免被惡意攻擊者竊取，而金鑰如果由單一伺服器保管它，可能會發生許多弊端，如：主管出賣了這把金鑰，導致機密資料流出去，造成資料的嚴重安全問題。因此，若將主金鑰打造成n份不同的次金鑰，分別由不同的雲端伺服器保管，便可以達到分散風險的目的，但此方法需要全體人員都到齊才有辦法解開秘密金鑰，比較不方便。除了把金鑰打造成許多不同的次金鑰之外，另外一個常見的雲端安全問題就是雲端伺服器的存取驗證，倘若有攻擊者偽冒雲端儲存伺服器，則使用者的資料可能會被攻擊者直接拿取，為避免偽冒雲端儲存伺服器的攻擊，最直覺的方式就是利用進行身份驗證機制，來防止此攻擊。

以目前所參考之文獻，大部分以單一的密碼系統來設計安全雲端儲存系統，故本研究將利用混合式密碼系統加密明文與金鑰，再加上門檻式機制將主金鑰分散，且伺服器可驗證其身分，以便讓使用者能安心使用雲端伺服器。

根據上述所提到的問題，過去已有許多學者提出相關研究來解決秘密分享的問題，例如：採用門檻式方法(Threshold scheme)，Shamir[6]的方法是將主金鑰隱藏於多項式 $f(x)$ 的常數項內。從 $f(0)=sk$ 可以輕易求得。每一位參與者都有唯一的公開識別碼 ID_i，莊家依據每位參與者的公開識別碼產生不同的子金鑰。每一 $(ID_i, f(ID_i))$ 可視為該多項式在二維空間上的一個座標，只有蒐集到至少t個以上座標才可以決定 $f(x)$，若少於t個座標，則訊息不完整，無法導出 $f(x)$，能達到完美安全。若知道t個子金鑰，便可利用拉格蘭治內插多項式(Lagrange Polynomial)回復主金鑰。

以上討論為分享單一金鑰(SK)，一旦被推導出來，SK便不能再被當成金鑰來分享，因為大家皆知道金鑰的內容了，下次便不需要再與其他人合作來導出SK，每個子金鑰只能使用一次、效率不高，且每個子金鑰的傳送都要非常謹慎且浪費時間。為了改善效率，Blakely[2][4]於1984年提出了傾斜方法(Ramp Scheme)，當知道d個以上的子金鑰時，可以推導出部分主金鑰，而知道m個子金鑰時，又能完全恢復主金鑰。

因應雲端儲存的蓬勃發展，2010年Lin[1][5]將門檻式方法應用在一個新的雲端儲存系統，結合門檻式公開金鑰密碼系統與容錯編碼去設計同時具有高度機密性與容錯能力之系統，同時提供安全的資料轉移機制，使用者可以將自己的資料授權給其他的使用者，除了加強儲存系統的安全性，也大大的提升系統的功能性，雖然安全性增加許多，但由於門檻式方法存在著2種角色，一個就是"參與者"，另一個就是"莊家"，而這兩種角色，皆可能成為"欺騙者"，而該系統並無提供驗證的功能，所以無法驗證伺服器所傳回的是否為真實的資料。

本文所提出的協定除了改善系統的效率之外，再加上驗證機制：先利用對稱式金鑰加密法對檔案加密，再透過ElGamal公開金鑰密碼系統(也可以利用其他種類之公開金鑰密碼系統)加密金鑰，雲端儲存系統中，許多檔案的加密通常在於初始階段，或者是將檔案更新，再重新加密上傳，相對於使用者在下載檔案、驗證、解密次數是最多的，但在其他方面卻更有效率；且增加了驗證機制，以進一步驗證伺服器是否已被惡意攻擊或惡意偽冒。

二、 相關研究

雲端儲存服務本身並不是一種新的技術，基本的功能不外乎就是上傳、下載，對一般使用者來說便足夠了，但對企業、公司或金融業來說，將敏感性資訊儲存在雲端伺服器，雖是省下了儲存空間的成本，但是資安防護的問題並不會因此消失，因此保存在第三方的資訊安全就變得非常重要，主要包括了資料的機密性、完整性、可用性。由於雲端運算技術近年來非常火熱，保護資訊的相關研究也越來越多，然而攻擊也不斷推陳出新，因此若惡意攻擊者得到雲端儲存系統中的資料，或在沒有任何使用者授權的情況下將資料隨意給予他人，都將洩漏了使用者的資料，除了破壞資料的機密性，也可能被惡意使用，造成使用者的危害。

回顧過去文獻中，Lin [5]將門檻式與雲端運算技術巧妙的結合，整合門檻式公開金鑰密碼系統與容錯編碼去設計具有高度機密性與容錯能力之系統，而該系統已被證明能

夠抵擋選擇明文攻擊(CPA)，Lin提出的系統模型[5]是基於Bilinear Map密碼系統，將每個檔案切割成同等大小區塊，各別加密後，再隨機儲存到使用者一開始選擇好的雲端伺服器群，而每一個伺服器會合併所有接收到的檔案，再選取k個隨機係數，將k個資料區塊與k個隨機係數作隨機線性組合，透過這個方式產生n個隨機線性組合，原先加密過的密文就會被編碼成資料區塊，若獲得至少k個線性組合與其相對應的係數，可將k個資料區塊解回，該系統可容許至多(n-k)個被編碼的區塊損壞或遺失。

本論文之雲端儲存系統除了用混合式密碼系統搭配門檻式方法，又將密文與系統所產生之矩陣進行運算，使其密文更加模糊化，其主要目的是應用在公司儲存機密文件於第三方時，可增加檔案的機密性與可用性，另外加上驗證的機制，以檢查伺服器是否為正常的狀態。同時我們也增加了一個金鑰伺服器，用來管理金鑰之拆解與回復，不授權給儲存伺服器，以防止伺服器之間互相勾結，造成使用者資料的機密性與可用性被破壞。

三、 基於混合式密碼系統之可驗證雲端儲存系統

3.1 符號說明

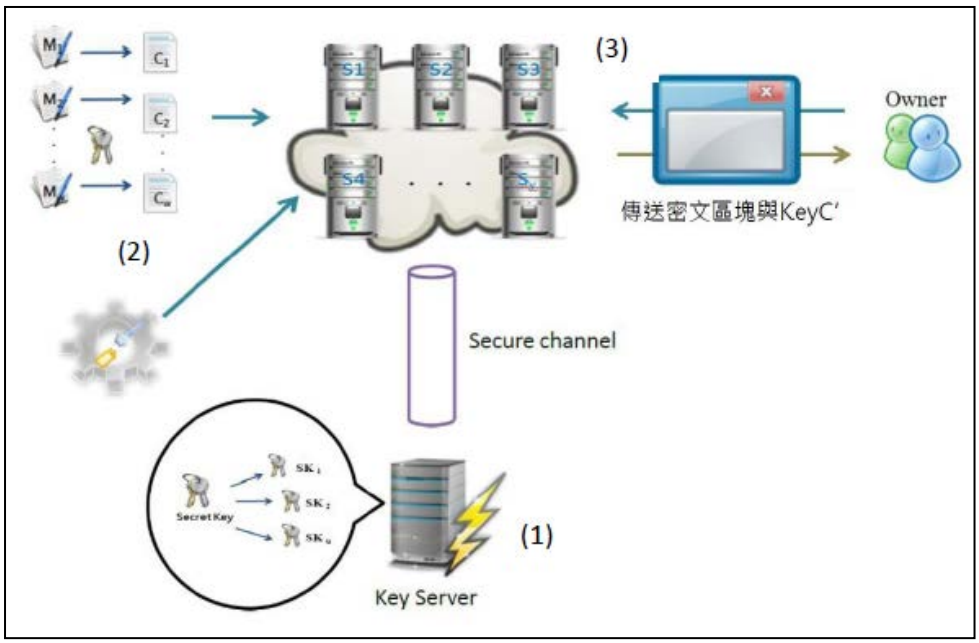
表一: 協定相關符號之描述

符號	說明
M	使用者的明文資料，假設 M 為一個完整的明文，根據 AES 加密法之規則分割成 u 個相同大小的明文區塊: $M_1, M_2, M_3, ..., M_u$ 。
C	使用者的密文資料，$M_1, M_2, M_3, ..., M_u$ 經加密過後所產生的密文區塊： $C_1, C_2, C_3, ..., C_u$ 。
$E_{A\_Key}(\sqcup)$	AES 加密演算法，金鑰為 A_Key。
$H(\sqcup)$	雜湊函數。
Owner	資料的擁有者。
S	儲存伺服器群： $S_1, S_2, S_3, ..., S_y$ ，負責儲存使用者的密文資料與子金鑰的伺服器。
KS	金鑰伺服器，用來產生 ElGamal 金鑰對與根據 S 來產生相對應之子金鑰。
ω	此參數為 AES 金鑰透過雜湊函數所產生的值。
KeyC	與 ω 做 Exclusive-OR 運算後產生之值，用來增加金鑰的安全性。
KeyC'	用 ElGamal 公開金鑰密碼系統加密後產生的參數。
t	門檻式方法之門檻參數。
n	子金鑰數目。

3.2 系統架構

我們設計了一個專門管理金鑰的金鑰伺服器，並將所需要的資訊透過一安全通道

傳給雲端伺服器，圖一為本文之系統架構。



圖一: 可驗證之門檻式雲端儲存系統架構

本系統主要分成三大階段：(1)設定階段、(2)加密階段、(3)存取階段：

(1) 設定階段:首先*KS*會產生ElGamal公開金鑰對，再將公鑰*PK*公開給使用者知道，私鑰*SK*以門檻式方式，根據參與伺服器之公開識別碼產生出相對應之子金鑰，再透過安全通道分別傳送給對應之伺服器。

(2) 加密階段:使用者會將明文切割成AES加密法規定之大小，再使用AES加密演算法將明文加密成密文區塊，並使用雜湊函數與Exclusive-OR運算加強AES金鑰之安全性，最後利用ElGamal系統之公鑰再加密AES金鑰一次，並與密文區塊一同傳送到伺服器。

(3) 存取階段:當使用者有需要時，透過一命令介面對伺服器下達下載檔案之指令，此時*KS*會指定一台伺服器當作主伺服器，每台Server會提交自己的 ID_i 、$sk_{i,j}$給主伺服器。主伺服器再將蒐集到的參數傳送給*KS*，透過驗證機制來驗證是否有伺服器為欺騙者或已被入侵。若通過驗證，便傳送true給主伺服器，將KeyC'解密，連同密文區塊一同回傳給使用者，讓使用者自行做最後的解密與合併明文。

本系統可分成六個演算法：

KeyGen ：一個產生公開、私密金鑰之演算法，由 *KS* 在設定階段執行。

ShareKeyGen ：*KS* 產生金鑰對之後執行此演算法，主要工作是將主金鑰分成 n 個子金鑰。

Ver ：*KS* 利用 H( )計算 *SK* 的識別確認碼，以便日後驗證。

Enc ：加密演算法，使用者用來對明文做加密動作。

ReCover ：*KS* 透過此演算法驗證伺服器是否為欺騙者或已被入侵。

Dec ：使用者從伺服器下載檔案後，透過此解密演算法解回金鑰與明文。

3.3 設定階段

*KS*首先使用**KeyGen**演算法產生本身金鑰對。

(1) 選取一大p，g為 Z_n^* 之原根

(2) 產生一隨機亂數：r

(3) 任選私鑰X，並計算公鑰：$Y = g^x \ mod \ p$

(4) 公開Y、g、p值

(5) 公鑰=Y；私鑰=X

接下來*KS*從Z_n^*中選出隨機亂數$a_1, a_2, \ldots, a_{t-1} \in Z_n^*$，並挑選*n*與*t*兩個參數， 再產生一個*t-1*階之多項式： $f(x) = (X + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1}) \bmod p$ ，再根據參與伺服器之公開識別碼 ID_i 產生對應子金鑰 $sk_i = f(ID_i)$，得到 (ID_i, sk_i) ，並刪除X，詳細演算法如下：

(1) *KS*從Z_n^*中選出隨機亂數$a_1, a_2, \ldots, a_{t-1} \in Z_n^*$，挑選*n*與*t*值，並產生
$f(x) = (X + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1}) \bmod p$

(2) 計算S_i的子金鑰： $sk_i = f(ID_i)$ ，$1 \le i \le n$ ，得到
$\{(ID_1, sk_1), (ID_2, sk_2), (ID_3, sk_3) \ldots (ID_i, sk_i)\}$

(3) 刪除主金鑰X

接著*KS*根據伺服器之公開識別碼來產生對應驗證值，以便日後驗證伺服器是否被入侵或遭攻擊，若為安全，才可以進行解密工作，詳細演算法如下：

執行**Ver** (ID,sk_1,sk_2,...,sk_i) →($sk_{i,j}$)

(1) *KS*產生H(・)，並依下列式子求得各*sk*的識別確認碼。

$$sk_{i,j} = \frac{H(sk_i)}{ID_j} \bmod p \quad , \quad j = 1, 2, \ldots, u$$

(2) 得到$\{(ID_1, sk_1, sk_{1,1}), (ID_2, sk_2, sk_{2,1}), (ID_3, sk_3, sk_{3,1}) \ldots (ID_n, sk_n, sk_{n,1})\}$,

$\{(ID_1, sk_1, sk_{1,2}), (ID_2, sk_2, sk_{2,2}), (ID_3, sk_3, sk_{3,2}) \ldots (ID_n, sk_n, sk_{n,2})\}$,

.

.

.

$\{(ID_1, sk_1, sk_{1,n}), (ID_2, sk_2, sk_{2,n}), (ID_3, sk_3, sk_{3,n}) \ldots (ID_n, sk_n, sk_{n,n})\}$,

(3) 透過一安全通道將ID_i、sk_i與$sk_{i,j}$傳送給S_i，$1 \le j \le n$

$\{ID_1, sk_1, sk_{1,1}, sk_{1,2}, sk_{1,3}, \ldots\ldots, sk_{1,n}\} \rightarrow S1$

$\{ID_2, sk_2, sk_{2,1}, sk_{2,2}, sk_{2,3}, \ldots\ldots, sk_{2,n}\} \rightarrow S2$

.

.

.

$$\left\{ID_n, sk_n, sk_{n,1}, sk_{n,2}, sk_{n,3}, ..., sk_{n,n}\right\} \rightarrow Sn$$

(4) *KS*保留每一台伺服器之$sk_{i,j}$識別確認碼。

3.4 加密階段

假設使用者有一明文M，欲上傳至儲存伺服器，先自行產生一把AES加密金鑰*A_Key*，將明文M分割成u個區塊，即 $M_1, M_2, M_3, ...M_u$ ，再用加密金鑰*A_Key*對u個明文區塊做加密動作，以下為**Enc**之演算法：

(1) 先將明文M分割成$\{M_1, M_2, M_3, ...M_u\}$ 。

(2) 利用金鑰*A_Key*及AES加密法計算$C_i = E_{A\,Key}\ (M_i), 1 \le i \le u$。

(3) $C_1, C_2, C_3, ...C_u$ 為加密後的密文區塊。

當密文產生後，*KS*再產生一矩陣$< b_{1,1}, b_{1,2}, ..., b_{u,u} >$，其值之區間為[1,*p*]，傳送給儲存伺服器，並將密文與此矩陣做矩陣運算，將資料編碼成：$C_1', C_2', C_3', ...C_u'$ 。

$$G=\begin{bmatrix} b_{1,1} & \cdots & b_{1,u} \\ \vdots & \ddots & \vdots \\ b_{u,1} & \cdots & b_{u,u} \end{bmatrix}$$

$$\left[C_1', C_2', C_3', ...C_u'\right] = G \times \begin{bmatrix} C_1 \\ C_2 \\ ... \\ C_u \end{bmatrix}$$

接下來，使用者再將AES金鑰加上該明文之檔名，利用雜湊函數H( )計算出：ω，再將AES金鑰與ω做XOR運算產生*KeyC*，此運算的目的主要是為了防止被不信任的伺服器自行運算出AES金鑰，而導致密文被洩漏出去，最後再使用*KS*之ElGamal的公鑰對KeyC加密，將最後的值KeyC′與密文區塊一併傳送至儲存伺服器，詳細演算法如下：

(1) 使用雜湊函數H(・)將*A_Key*與該檔案之檔名計算出ω，再將ω與*A_Key*做XOR運算後，算出*KeyC*後，再使用ElGamal之公鑰對*KcyC*加密成 $KeyC'$ 。

$$\omega = H(A\_Key \,\square\, FileName)$$
$$KeyC = A\_Key \oplus \omega$$
$$KeyC' = E_{pk}(KeyC)$$

(2) 並將 $KeyC'$ 儲存到儲存伺服器中。

$$KeyC_i' = (ID_i, g^{r \cdot sk_i}, keyC \cdot pk^r), i = 1, 2, 3.....u$$
$$\left\{KeyC_i', _{,i=1,2,3....u}\right\} \rightarrow S$$

(3) 將$C_1', C_2', C_3', ...C_u'$傳送給*Server*群。

3.5 存取階段

當使用者欲存取檔案時，會要求伺服器傳回密文區塊與金鑰密文，且驗證確實為伺服器所傳送，並非是由假冒的攻擊者所傳送。首先，*KS*會指定一台主伺服器，而*t*台伺服器會提交自己的ID_i、$H(sk_i)$給使用者。主伺服器再將蒐集到的參數傳送給*KS*，透過一

驗證式來驗證身分，檢查是否有假冒或已被入侵。若皆通過驗證，便傳送TRUE給主伺服器以便回覆給所有伺服器，並開始做部份解密之動作，否則傳回FALSE，並告知使用者該伺服器已被入侵。將 $KeyC'$ 解密，連同密文區塊一同回傳給使用者，讓使用者自行做最後的解密與合併明文。主伺服器會驗證Server群傳送的值是否是真實的，若不真實，則停止回覆。下列為演算法：

$$\mathbf{Recover}(t, sk_1, sk_2, \ldots, sk_t) \rightarrow sk$$

$$sk_{i,j} \stackrel{?}{=} \frac{H(sk_i)}{ID_i} \bmod p$$

若驗證正確，回傳" TRUE"，若不正確，回傳"FALSE"。若回傳的值是正確的，儲存伺服器利用反矩陣將編碼過後的密文解回：$C_1, C_2, C_3, \ldots C_u$ ，將 $KeyC'$ 做部份解密，再與C_i一同傳回給使用者，使用者再自行把正確的金鑰解開，並解密密文及自行合併。相關演算法如下：

(1) 儲存伺服器會用反矩陣解碼密文。

(2) 蒐集t個 $KeyC_i'$ 。

(3) 收到回傳"TRUE"的t台儲存伺服器會計算 z_i 值： $z_i = \prod_{\substack{i=1 \\ i \neq i}}^{t} \frac{-ID_i}{ID_j - ID_i} \bmod p - 1$

(4) 將t個 $KeyC_i'$ ， $g^{r \cdot sk_i \cdot z_i}$ 與密文區塊一起傳回給使用者。

$$\left\{ KeyC_i', g^{r \cdot sk_i \cdot z_i}, C_{i,i=1,2,,3\ldots u} \right\} \rightarrow User$$

(5) 由使用者自行將t個 $KeyC_i'$ 解回 $KeyC$ 。

$$KeyC' = KeyC_i' \cdot g^{\sum r \cdot ski \cdot zi} \bmod p$$

$$= pk^r \bmod p$$

$$KeyC = keyC \cdot pk^r \cdot pk^{-r}$$

(6) 計算 ω ，並與 $KeyC$ 做XOR運算，將 A_Key 解回，並對 C_i 解密

$$\omega = H(A\_Key \square FileName)$$

$$A\_Key = \omega \oplus KeyC$$

$$M_i = D_{A\_Key}(C_i), \quad 1 \le i \le u$$

(7) 最後，再把所有的 C_i 合併解密回完整的明文 M 。

$$M = \{M_1, M_2, M_3, \ldots M_u\}$$

四、 安全性分析

安全性分析的部分，我們將以機密性、完整性及確認性三方面來分析：

(1) 機密性

使用者針對已分割過的檔案，用私鑰密碼系統進行加密，且再透過一雜湊函數對AES金鑰計算，再使用ElGamal系統之公鑰對AES金鑰做加密動作，由於系統安全性植基於離散對數難題，若惡意第三方得到了 g , Y , p，也難以算出私密金鑰，且在金鑰伺服器上採用門檻式的方法將私鑰拆成多把子金鑰，可以增加攻擊的困難度。

(2) 完整性

本系統對明文進行加密後，會再加入一矩陣，利用矩陣運算編碼成新的資料，使其資料再更加安全外，在進行解碼時，反矩陣計算出來的值與原矩陣之值不相等，則表示資料已不具備完整性，若相符，則表示資料的完整性仍存在。

(3) 確認性

由於門檻方法中存在著欺騙者的問題，所以在存取階段中，*KS*會對儲存伺服器傳回之驗證參數與當初所產生的驗證參數值進行比對，以確保伺服器之間沒有勾結、欺騙等行為或被入侵。

五、 效能分析

本文採用ElGamal系統與AES系統混合的方式來保護雲端資料之安全性，其中ElGamal系統可以用其他公開金鑰密碼系統取代，AES系統也可以用其他秘密金鑰密碼系統取代，並無衝突，本文只是透過此兩系統予以說明整體做法。而使用混合式系統的優點不外是可以提升其效能並確保安全性。表二為所使用相關密碼系統之複雜度比較表，由表三及表四可發現，本文新提之方法雖然在金鑰加密運算上佔了相當大的運算量，但在解密時，本文之協定卻是比Lin [5]來的有效率。

六、 結論

現今已有許多使用者將資料外包儲存在雲端的伺服器上，以減少本機儲存的空間和維護上的成本，除了上述兩樣效益之外，有些雲端儲存伺服器也會利用加密法將資料加密保護，以達到資訊安全中的機密性、完整性與可用性。本文研發出將門檻式加密法應用至雲端儲存伺服器，且整合 ElGamal 密碼系統與 AES 密碼系統，以保護雲端儲存資料的安全性。因本系統採用 AES 密碼系統對檔案進行加密與解密，因此，在檔案加解密速度方面較先前文獻來的有效率。此外，本系統採用混合式密碼系統，整體系統將更有效率。

表二: 計算量倍率表

計算方式	敘述	計算量比率
P	Pairing運算	200000
H	雜湊函數	30
Ae	AES加密運算	200
Ad	AES解密運算	100
Add(G)	點加法運算	26
Mul( )	整數乘法運算	1
Mod( )	整數模數運算	10
XOR	互斥或運算	0.5

表三: 傳輸量表

Operations	文獻[5]	本論文
密碼系統	Bilinear Map	AES + ElGamal
訊息加密 (k 訊息值)	k P + 2k E(G_1) + k Mul(G_1)	k Ae(Zp)
金鑰加密 (for AES key)	×	1H + 1XOR + 1Mul(Z_p) + 1 Mod(Z_p)
解碼	k^2E(G_1) + (k-1)k Mul(G_1) + O(k^3) Fp	u個XOR
解密	t E(G_1)	k Ad(Z_p)
伺服器傳輸量 (for *KS*)	u把secret key子金鑰	u把secret key子金鑰
使用者傳輸量	k P	k Ae(Z_p) + 1把AES加密金鑰
驗證	×	1H+2Mod( )

表四: 計算總量表

Operations	文獻[5]	本論文
密碼系統	Bilinear Map	AES + ElGamal
訊息加密 (k 訊息值)	200,028	200
金鑰加密 (for AES key)	×	43
解碼	28	0.5
解密	200,000	100
驗證	×	50

謝啟

本文感謝審查委員寶貴意見，與科技部計畫 MOST 104-2221-E-218-016 之補助，特此致謝。

參考文獻

[1] 林孝盈, “探討雲端儲存系統之資料機密性與功能性共存證明”, Comm. Of the CCISA, Vol.18, No.2, Apr. 2012:pp.35-45.

[2] 賴溪松、韓亮、張真誠, “近代密碼學及其應用”, 旗標出版股份有限公司, 台北市2003.

[3] Behrouz, A., Forouzan, “Introduction to Cryptography and Network Security,” The McGraw Hill Companies, 2008.

[4] Blakley, G.R., Catherine Meadows, “Security of Ramp Schemes,” 1984:pp. 242-268.

[5] Lin, H.Y., “Data Confidentiality and Robustness in Decentralized Cloud Storage Systems”, Dep. of Computer Science NCTU, May 2010.

[6]Shamir, A., “How to Share a Secret”, Comm. of ACM, Vol. 22, 1979:pp. 612-613.