驗證檔案伺服器輸出之正確性

許家瑞
中興大學
資訊科學與工程學系
研究生
e-mail：
s9656038@csmail.nchu.edu.tw

洪國寶
中興大學
資訊科學與工程學系
教授
e-mail：
gbhorng@cs.nchu.edu.tw

摘要

搜尋加密資料在資訊發達的時代中，已經成為非常重要的部分，尤其人們將隱私資料存放於開放式網路環境的伺服器，使得保護資料隱密性越來越重要。然而，大部分的方法都專注在探討搜尋加密資料內容之正確性與效率，卻忽略了驗證伺服器回傳資料數量與是否為更新後文件之重要性。本文提出測試法，解決伺服器回傳資料數量之正確性和檢查是否為更新後文件。測試法則是利用建立關鍵字與文件編號對映關係之表格，當伺服器回傳搜尋結果時，可驗證此伺服器是否異常。此架構在驗證時之時間、通訊量、記憶體需求皆可符合簡易電子裝置之需求。

關鍵詞：關鍵字搜尋、測試、驗證

Abstract

Nowadays, protecting personal private information is getting more significant than before. We hope that the private information in the open network server could be accessed from anywhere, at anytime. Furthermore, if the servers or attackers modified any information or retrieved amount is wrong. We can check it. In this paper, we propose a scheme to verify that whether the server can be trusted or not. The memory and communication cost of the scheme conform to the mobile devices.

Keywords: **Keyword Search, Test, Verify**

1. 前言

對於網路上伺服器的安全，一般來說是假設在誠實且可信任的伺服器下討論，並且維護其安全為前提做研究。但隨著資訊科技的發達以及惡意使用者的增加，對於系統設計上面的瑕疵也漸漸的被發現。在惡意使用者得到這些資訊後，往往利用這些系統的弱點來加以攻擊。舉例來說，使用者將本身資料儲存在信任的伺服器端，伺服器端可以提供驗證使用者的身份機制，所採用的是以帳號與密碼為控制安全存取之方法。但是，對於使用者驗證伺服器端的議題，卻是少有人探討。

現今的使用者會將重要的資料放置於網路伺服器上，目的是為了備份以及達到資料可攜性。使用者首先相信，具有信任力與具有信用的伺服器端。不過，一旦所信任的伺服器端出了問題，之後就會發生極大的損失。例如：(1)伺服器端已被攻擊者假冒，攻擊者隨意竊取使用者資料來犯罪。(2)伺服器端所做的行為不誠實，像是竄改資料內容或數量不正確。例如：對於使用者所儲存在伺服器端的資料，當使用者向伺服器端提出文件搜尋，伺服器所提供的資訊文件卻不完全、或者伺服器提供給使用者錯誤的文件。情境像是攻擊移民署的非法出入境名單、警政系統的管束或罪犯名單、金融業的交易資料或醫院的病歷資料…等，攻擊者的目地可能不在竊取資料，而在竄改內容或相關文件的數量上，讓攻擊者取得一些利益，像海關闖關成功、或順利通過臨檢、交易數量被變更或病歷竄改等。以上的攻擊無法使用帳號密碼或者是防毒軟體、防火牆等安全措施來防護，此種攻擊方式已經不再對具有信任力的伺服器端發出攻擊，而是針對使用者端；由於使用者本身的安全策略在架構時的成本比起企業級架構的成本少了許多，因此在安全防護的等級也稍嫌弱勢。這對使用者來說，對於偽造伺服器之欺騙的攻擊者其實防不勝防，因此，使用端如何驗證伺服端之研究已愈行重要。

我們針對此方面的弱點探討且提出解決方案。首先假設伺服器端是不可信任，因為伺服端可能被偽造，所以需先證明伺服器為安全可信任後再放心使用。本文所要探討的，就是針對伺服器端的行為作分析，進而確認伺服器端是否為誠實且可信任或者為攻擊者假冒模仿。不只是伺服器端來驗證使用者的身份，而

是使用者也可以同時測試伺服器端的身份，並且確認伺服器的行為是否可信任，以達成雙方互相信任，透過自我主動的防護強化網路使用的安全性。

假如有一位使用者叫 Alice，她想將文件放置在不可信任的伺服器上。我們把這不可信任的伺服器叫 Bob，因為文件不希望被知道，所以一定是加密後的資訊，才上傳至伺服器端，在這部分跟 Song 等人[9]假設一樣，然而，我們要的不只是搜尋回來的文件是我們要的，並且我們還能藉由傳送關鍵字與回應的過程中，就能判斷伺服器是不是有少傳送符合的文件，或是傳送錯誤的文件，進而偵測伺服器是否異常，可讓使用者決定是否要停止傳送檔案給伺服器。

本文非只討論關鍵字搜尋技巧，而是針對搜尋過程，須達到執行資料搜尋是完整且正確。其中，完整性與正確性指：使用者端要驗證所有由伺服器回傳結果的完整與正確。

然而在不信任環境下，如何得知伺服器有無異常呢？所以本文先定義驗證伺服器的方式如下：

(1) 內容驗證(Integrity Verification)：在伺服器經過搜尋之後，將回傳文件，最後由使用者驗證資料內容的正確性。
(2) 數量驗證(Quantity Verification)：在使用者端，已知文件數量，且內容驗證無誤，搜尋已知文件的相同關鍵字，在伺服器經過搜尋之後，將回傳文件，最後由使用者驗證數量是否正確。
(3) 更新驗證(Update Verification)：在使用者端，已修改文件上傳至伺服器，然而經過伺服器搜尋後，所回傳文件，為修改之後的文件，而非前一版的文件。

如果達成以上三個驗證，則伺服器目前並無異常。在內容的驗證方面，目前的技術有 HASH、MAC、HMAC、數位簽章(Digital Signature)等技術。第二部分是介紹在加密關鍵字搜尋相關研究，也進一步說明驗證內文並不難，但驗證回傳文件數量的正確性是一很大的挑戰。因此，第三部分本文就討論在數量的驗證方面，我們提出的方法，利用偽造的方式來式，來達到內容與數量驗證。第四部分將分析我們的方法，第五部分則是比較與討論相關優缺點。第六部分將總結我們未來要努力的方向。

2. 相關研究

我們定義何謂誠實，好奇，不可信任(un-trust)的伺服器，誠實是可以正確的回應或直接執行比對動作，好奇就是在執行的同時會想要獲得文件內容的相關資訊，若在資料查詢的過程中，只要惡意的欺騙，包括更改資料或回傳不實的資訊，即為不可信任之伺服器。亦即本文假設伺服器端為不可信任的伺服器，因資料有可能被竄改，且本文重點是在於資料是否被竄改，與回傳文件數量是否正確。如果被竄改資料或傳送文件有錯，則透過本文所提出之方法，可偵測出伺服器傳送錯誤資料。即伺服器的異常，且不排除是遭駭客入侵竄改，或程式出錯所造成的。

另外，在 Artzi 以及 Song 等人[1, 9]提出的加密文件搜尋機制定義了四項加密搜尋所需遵守的特性：

(1) 控制搜尋(Controlled Searching)：伺服器端在沒有使用者的認證下無法進行有意義的搜尋。
(2) 獨立詢問(Query Isolation)：伺服器端無法由搜尋結果得到其他未搜尋的資訊。
(3) 隱藏搜尋(Hidden Queries)：伺服器端無法根據關鍵字的暗門(Trapdoor)猜出關鍵字為何。
(4) 獨立更新(Update Isolation)：使用者更新上傳文件時，伺服器端無法得到其他資訊。

前四項特性主要是讓誠實且好奇伺服器端在沒有使用者端的允許下，無法自行搜尋、取得加密檔案之資訊。而本文所提出的架構是建立在將資料文件寄送於不可信任的伺服器端上，所以除了滿足以上四項特性外，還必須滿足下列的第五項要求。

(5) 正確性測試(Correctness Test)：使用者傳送文件至伺服器時，伺服器無法分辨正常文件與更新資訊的文件，也無法從中得到其他資訊，然而，在經過測試之後，能正確的傳回有符合文件即可。

然而，在加密文件搜尋機制上[1-7, 9]，由 Song 等人[9]先提出 SWP 對稱式加密方式。首先將文件加密後上傳至伺服器，接著使用者傳送暗門(Trapdoor)給伺服器，讓伺服器利用暗門對文件進行搜尋，當搜尋到相同的時候再將文件回傳給使用者。之後的相關研究有 Goh[3]、Boneh[2]，而 Goh 利用 Bloom Filter 來達到快速加密與搜尋，Boneh 等學者，則是利用公開金鑰密碼系統來建構加密文件搜尋。再來說明 Sion[7]等學者的方法，則是提

出利用檢查和(Checksums)與擬亂函數(Pseudo-random Function)，用來檢查查詢的資料是否正確。

在 Sion[7]等學者的方法內(簡稱 SC scheme)，一開始先選 2 個祕密值 p 和 x，而 p 是一個大質數，且 x 是一個隨機整數(random integer)介於 $1 < x < p$。建立一個關鍵字文件集(Keyword Document Sets, KDS)，公式如下：

$$\prod_{i=1}^{t}(d_i + x) \bmod p$$

圖 1 描述 SC scheme 在 Checksums 時之運算。其關鍵字與文件的對映關係的圖 1 所示。

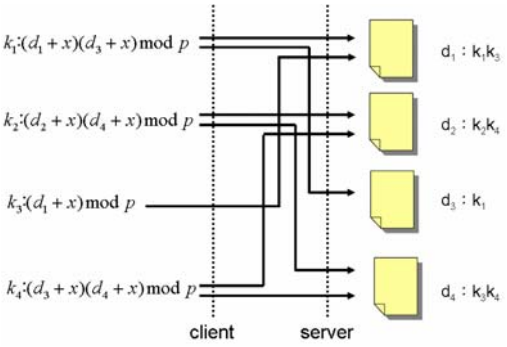

圖 1 SC scheme Checksums 架構

即藉由關鍵字可推得回傳之文件數量是否是正確的。圖 2、圖 3 描述 SC 方法如何打亂資料後上傳至伺服器。

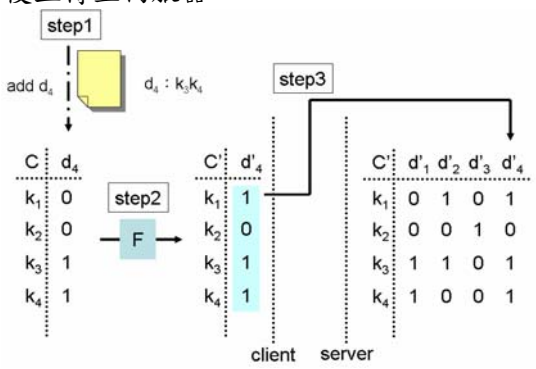

圖 2 SC 擬亂函數架構

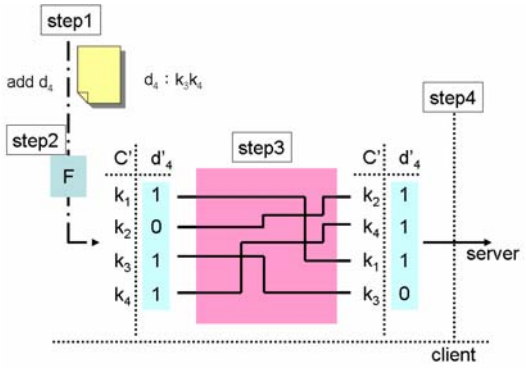

圖 3 SC 重新排列架構

圖 2，首先在原有陣列中加入 d_4 文件，對應到關鍵字的陣列中，然後 step2 是一個可逆的擬亂函數，且此可逆函數是一個擴大函數，即其中有包涵偽造的資訊。接著圖 3，上傳 C'(偽造表格)的表給伺服器前，step3 是一個打亂 k 值的方式。所以上傳給伺服器的表格為一個偽造表格，用來驗證伺服器是否異常。最後說明整個驗證的原理，如圖 4 所示。

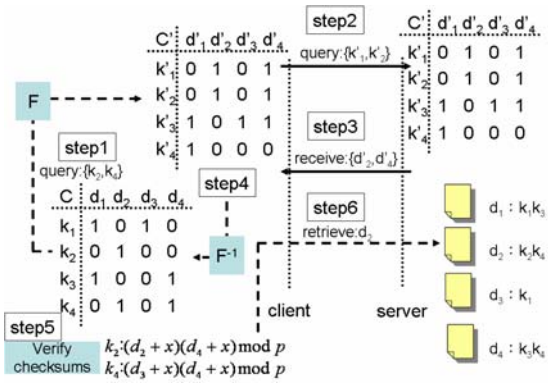

圖 4 SC scheme 例子

建立 C 為關鍵字對映文件的表格，經過擬亂函數 F 建立 C'表格，在將 C'與密文上傳至伺服器。最後說明查詢的過程，setp1 一開始將 query 的 k 值經過擬亂函數 F，找出對映的 k 對伺服器做查詢，再由伺服器比對搜尋之後回傳結果為 d'_2 與 d'_4，其中包括偽造之文件資訊。在 client 端接收之後可由 C'比對出對映文件是否正確，而後在 setp4 經過逆運算 F^{-1}，比對表格 C 與驗證 k_2 與 k_4 的值，即可得知 d_2 為我們要的文件而 d_4 為偽造之文件。最後 setp6 取回 d_2 文件。

在對稱式加密則由 Eun-Kyung Ryu[6]等學者，提出 CKSE 方式來完成加密關鍵字搜尋。

CKSE 方法首先定義三個 size 為 p 的群，G_1 、G_2、G_T 為循環群，g_1 屬於 G_1 且是 G_1 的生成元，g_2 屬於 G_2 且是 G_2 的生成元。我們需要一個任意值映射到點的雜湊函數 H:{0,1}*→G_1，以及一個雙線性映射(bilinear map) e: $G_1 \times G_2 \rightarrow G_T$，雙線性映射擁有以下性質：

(1) 可計算性(computable): 任意兩個點 g 屬於 G_1，h 屬於 G_2，有存在一個多項式時間的演算法 e，來計算 $e(g,h)$屬於 G_T。

(2) 雙線性(bilinear): 對於 1 到 p 的任意整數 x、y，則 $e(g_1^x, g_2^y)=e(g_1, g_2)^{xy}$。

(3) 不對退化性(non-degenerate): 如果 g_1 為 G_1 的生成元，且 g_2 為 G_2 的生成元，則 $e(g_1, g_2)$ 必為 G_T 生成元。

CKSE 機制主要由四個演算法構成。

(1) KeyGen(1^k): 產生收件者的金鑰(key)，且 key 屬於 G_1，g_2 則不公開。

(2) Enc(*key, D*): 寄件者利用收件者私鑰對關鍵字集合 *D* 產生可共節搜尋的密文 C_i。選一隨機數 α_i 屬於 Z_p，並取得關鍵字對應到的雜湊值，得到：

$$C_i = \left(e(key, g_2^{\alpha_i}), g_2^{\alpha_i}, H(W_1)^{\alpha_i}, ..., H(W_m)^{\alpha_i}\right)$$

(3) Trapdoor(*key*, W_i): 收件者利用欲搜尋的內容與金鑰產生暗門 T_w。選一隨機數 β 屬於 Z_p，得到暗門 T_w 為 $[T_1, T_2, I_1, I_2, ..., I_t]$，其中

$$T_w = \left(key \prod_{t=1}^{l} (H(W_t))^{\beta}), g_2^{\beta}, I_1, I_2, ..., I_t\right)$$

而 $I_1, I_2, ..., I_t$ 為文件中的欄位值。

(4) Test(T_w, C_i): 驗證等式

令 $T_w=(T_1, T_2)$ 和 $C_i=(V_1, V_2, C_{i,1}, \ldots, C_{i,m})$ 且 i=1~n，其關係為下：

$$V_1 = \frac{e(T_1, V_2)}{e(\prod_{t=1}^{l} C_{i,jt}, T_2)}$$

若比對相等則輸出"yes"為有搜尋到符合的關鍵字，否則輸出"no"。驗證等式過程如下：

$$e(key, g_2^{\alpha_i}) = \frac{e(key \cdot (H(W_t)^{\beta}), g_2^{\alpha_i})}{e(\prod_{t=1}^{m} (H(W_t)^{\alpha_i}), g_2^{\beta})}$$

在眾多加密關鍵字搜尋的方法中，我們採 CKSE 來說明其運作原理，但是這些方法都只有做到驗證文件中，是否有對映到搜尋的關鍵字，而且沒有考慮到回傳回來文件的數量，若數量不對，則如何找出來呢？另外，值得一提的是更新之後的文件，在上傳之後，如果不可任信的伺服器，只傳送更新前的文件，就在更新上是有問題的，然而本文所提的方法，就依這系統架構下，提出可以解決驗證數量與文件內容正確性的方法。

3. 我們的方法

本文的基本構想是要儲存一個對照表，以利之後要來做數量的驗證，然而上傳偽造文件，是為了拿來驗證是否為更新後的文件，所以在系統架構之後，先說明我們如何建表解決數量驗證的問題，以及之後解決更新驗證的方式。

3.1 系統架構

使用者寄出文件的訊息為下：

$$C_i = \left(E_{key}(ID_i \| h \| \alpha_i), e(key, g_2^{\alpha_i}), g_2^{\alpha_i}, H(W_1)^{\alpha_i}, ..., H(W_m)^{\alpha_i}\right)$$

其中 *key* 為使用者的金鑰，ID_i 為文件編號，且文件編號是不重復，i=1~n，n 為文件的個數，$h=HMAC(W_1||W_2||...||W_m)$，$i$=1~$m$， h 為所有關鍵字做訊息認證碼後的值，m 為關鍵字的欄位(field)數，而 α_i 屬於 Z_p 一個均勻分佈的隨機數，H 為關鍵字加密演算法，並且不會洩漏任何有關內文的資訊，卻能夠搜尋不同的關鍵字，採用 Golle 等人[4]的假設：

(1) 同一份文件中，相同的關鍵字不會出現在兩個不同的欄位；就是同一文件內欄位上的關鍵字不會重複出現。

(2) 每份文件都定義每個關鍵字欄位。

如果文件是所謂的電子郵件的話，我們可以舉一個簡單的例子，假設定義了三個欄位：From、To、Date，可以把電子郵件的關鍵字表示如下圖 5。

m+3個欄位

n份文件	Document	V_1	V_2	From	To	...	Date
D_1	$E_{key}(ID_1 \| h \| \alpha_1)$	$e(key, g_2^{\alpha_1})$	$g_2^{\alpha_1}$	$H(W_1)^{\alpha_1}$	$H(W_2)^{\alpha_1}$	...	$H(W_m)^{\alpha_1}$
D_2	$E_{key}(ID_2 \| h \| \alpha_2)$	$e(key, g_2^{\alpha_2})$	$g_2^{\alpha_2}$	Alice	Bob	...	2008/12/11
	...	...	...	...	...	...	...
D_n	$E_{key}(ID_n \| h \| \alpha_n)$	$e(key, g_2^{\alpha_n})$	$g_2^{\alpha_n}$	$H(W_1)^{\alpha_n}$	$H(W_2)^{\alpha_n}$	...	$H(W_m)^{\alpha_n}$

圖 5 電子郵件格式之範例

也就是當使用者丟暗門給伺服器時，伺服器會從 From 之後，依選定的欄位開始比對，比對到相同的關鍵字之後，將文件回傳至使用者。

3.2 測試方法(Test Scheme)

測試方法，分二部份，建表、測試。

建表方式：

利用文件編號(ID_i)加總，來達到記錄數量正確性，關係如下：

$$S = \sum_{i=1}^{t} ID_i \bmod q$$

其中 S 為對映關鍵字的文件編號加總再 mod q 的值，而 ID_i 是文件編號，假設有 1 到 t 份有對映到的文件，q 為一個大質數。再來說明的部份，是我們僅存所有的關鍵字，我們稱為關鍵字集(Keyword Sets, KS)與對映總數(S)。如圖 6 所示。

KS		S
field number	keyword	

圖 6 表格格式

由圖 6 中，就可以得知 *KS* 欄位是放欄位編號與關鍵字，而 S 為對映關鍵字文件編號總和後 mod q，再來討論如何更新、新增與刪除。

(1) 更新：將修改後的文件，給予新的文件編

號，更改原對映加總的關鍵字關係，若有對映到的關鍵字，其 S 加上新的文件編號，若原本有對映的關鍵字，經過修改後，沒對映到，則要對 S 減去該文件編號。

(2) 增加：對欲新增的文件，在表格內搜尋對映的關鍵字，比對到對映關鍵字時，對 S 增加新的文件編號。

(3) 刪除：對欲刪除的文件，在表格內搜尋對映的關鍵字，比對到對映關鍵字時，對 S 刪除該文件編號。

所以以上就利用建立表格，就可以達到數量上的驗證，亦可驗證文件內容的正確性，與更新的驗證。以下列測試部份，亦可說明如何驗證。

測試部份：

透過修改文件後，將文件上傳至伺服器，來測試伺服器是否更新文件資訊，步驟如下圖 7 所示。

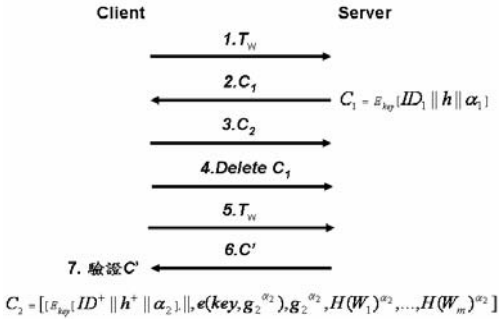

圖 7 測試步驟

(1) 使用者將關鍵字利用暗門包裝後，傳送至伺服器。

(2) 伺服器利用暗門比對搜尋後，回傳對映文件。

(3) 將原密文 C_1 修改之後，為密文 C_2，其中 ID_1 與 ID^+ 為不同的文件編號，上傳至伺服器端。

(4) 刪除原密文 C_1。

(5) 使用者將關鍵字利用暗門包裝後，傳送至伺服器。

(6) 伺服器利用暗門比對搜尋後，回傳對映文件 C'。

(7) 使用者驗證回傳文件的數量、內文正確性外，若有回傳到更新前的文件，則因為文件編號不重覆性，讓使用者可以驗證利用表格可驗證伺服器是有異常。

然而前 4 步驟是一般更新就一定尊循的步驟，所以是每一次的查詢，都可以測試伺服器是否有異常。

4. 安全性分析

關於本文所提出的方法，安全性分析分成二部份，一為文件內容的分析，二為文件數量的分析，然而，在文件內容的分析上，就這方面的技術，已有 HASH、MAC、HMAC 和數位簽章等基本驗證技術，可從 Stallings[10]等教科書，找到相關資訊，這部分我們就不分析討論，再來就針對文件格式加密後，利用關鍵字搜尋等技術，則在[1-7,9]等相關文章有討論到，對於這方面的安全性證明，大多都是說明解 Decision Diffie-Hellman Problem(DDH):訂定 Z_p 生成子（generator）g，p 為大質數，DDH 問題是很難於分辨兩組數 (g^a,g^b,g^{ab}) 與 (g^a,g^b,g^c)，其中 a,b,c 為 $\{1,\ldots,p\text{-}1\}$ 中的隨機值。我們說有一個有多項式時間運算能力的對手 A 具 ε 能力去解 DDH，寫成 $|\Pr[A(g^a,g^b,g^{ab})=true]-\Pr[A(g^a,g^b,g^c)=true]|>\varepsilon$，若 ε 是可忽略的則說 DDH 是困難的。然而證明其安全性，可參考 Ryu 等學者[6]的文章。

再來我們說明伺服器有那些攻擊方式：

(1) 藉由使用者傳送的暗門或檔案，分析是否要測試伺服器。

(2) 藉由放至在伺服器上的文件，在搜尋關鍵字時，只回傳了部份對映的文件、錯誤的文件，或不傳文件，讓使用者資訊更少或不正確。

從我們的方法中，就可以簡單的避開上述的二種攻擊方式，原因是因為使用者建立關鍵字與文件編號對映關係的表格，而且因為我們採用的文件編號是不重覆性，也就是更新前與更新後的文件，文件編號不會相同，由此區分出伺服器回傳的文件，是否為更新前的文件，所以我們的方法在數量與更新上都有支援。

碰撞機率：

因為在使用者這方，建立了一個關鍵字與文件的對照表，所以一定要壓縮其值，也就是在我們所提的方法中，mod q，q 為一個大質數，然而在碰撞機率的評估上為 $1/q$。

5. 比較與討論

就記憶體來說，關鍵字與文件的關係，若有 n 份文件與 m 個關鍵字，最大容量為 $O(n\cdot m)$。也就是建一個表格最大容量為 $O(n\cdot m)$。所以伺服器端必須要儲存所有的文件 n 與文件中所有的關鍵字 m，所以為 $O(n\cdot m)$，最後我們把本文的方法整理成表 1。

使用者端記憶體消耗成本：

在 SC 的方法中，因為所存的表格為關鍵字(k)與文件數量(n)的關係，所以成本為$O(k \cdot n)$，然而測式方法中，在測試伺服器時，必須先存下偽造的相關資訊，以利將來搜尋時，才能比對是否異常，假設關鍵字為k個，則成本為$O(k)$。

驗證：

驗證上，我們會去看回傳數量的正確性與內容的正確性，而內容的正確性，又可以討論是否有更新該份文件呢？所以我們把這驗證稱為更新驗證，然而在 SC 方法中，並沒有說明到更新驗證這方面的問題。

6. 總結

本文針對伺服器端的行為作分析，進而確認伺服器端是否為誠實可信任或者為攻擊者假冒模仿。不只是伺服器端來驗證使用者的身份，而是使用者也可以同時測試伺服器端的身份，並且確認伺服器的行為是否可信任，以達成雙方互相信任，透過自我主動的防護強化網路使用的安全性。然而驗證伺服器的方法也不只這一種，未來則是期望能降低各方面的儲存量，並且在驗證運算上更快更有效率，依然可達驗證回傳資料之正確性。

致謝

感謝匿名論文審查委員對本篇論文的建議。本研究由國科會補助完成，計畫編號：NSC96- 2628-E-005-076-MY3。

參考文獻

[1] S. Artzi, A. Kiezun, C. Newport and D. Schultz, "***Encrypted Keyword Search in a Distributed Storage System***," Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, 2006.

[2] D. Boneh, G.D. Crescenzo, R. Ostrovsky and G. Persiano, "***Public Key Encryption with Keyword Search***," Proceedings of IEEE Symposium on Security and Privacy, pp.44-45 IEEE, Apr 2004.

[3] E-J. Goh, "***Secure Indexes***," The Cryptology ePrint Archive, Report 2003/216, Mar 16, 2004.

[4] P. Golle, J. Staddon, and B. Waters, "***Secure Conjunctive Keyword Search over Encrypted Data***," Proceedings of Applied Cryptography and Network Security Conference, LNCS 3089, Springer-Verlag, pp.31–45, 2004.

[5] L.T.A. Joseph, A. Samsudin and B. Belaton, "***Efficient Search on Encrypted Data***," Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication, pp.352-357, Nov 2005.

[6] E-K Ryu, Takagi, T. "***Efficient Conjunctive Keyword-Searchable Encryption***," Advanced Information Networking and Applications Workshops, 2007

[7] R. Sion and B. Carbunar, "***Conjunctive Keyword Search on Encrypted Data with Completeness and Computational Privacy***," The Cryptology ePrint Archive, Report 2005/172, Jun 10, 2005.

[8] CHAPTER: Radu Sion, "Towards Secure Data Outsourcing," in "***The Handbook of Database Security: Applications and Trends***," M. Gertz and S. Jajodia (editors), Springer Verlag 2008.

[9] D. Song, D. Wagner and A. Perrig, "***Practical Techniques for Searches on Encrypted Data***," Advances in Cryptology - EUROCRYPT 2004, pp.506-522, May 2000.

[10] Stallings, W. "***Cryptography and Network Security: Principles and Practices***," Prentice Hall, Fourth Edition, 2007.

表 1 方法之比較

	SC scheme	我們的方法
使用者端記憶體消耗成本	$O(k \cdot n)$	$O(k)$
數量驗證	Yes	Yes
內容驗證	Yes	Yes
更新驗證	—	Yes