改善Song 等人之加密文件搜尋後使用者解密的效率

紀銘偉 洪國寶

國立中興大學資訊工程所

s9556027@cs.nchu.edu.tw

摘要

在不信任伺服器的狀況下，欲將文件放置於伺服器上，使文件能被搜尋但卻不想洩漏任何文件資訊，可以將文件加密並創造某些只有使用者自己知道的暗門 (trapdoor)。將來想取回含有特殊關鍵字的文件時，只要將此暗門給伺服器則可取回使用者想要的文件。如此一來可減少使用者儲存空間的負擔，並可隨時隨地取得重要文件，如果沒有使用者的授權將無法得到關於文件的任何資訊，即使在搜尋時伺服器也無法得到有關於文件的資訊，伺服器只能得到我們允許被得知的資訊，此資訊無關於文件的內容。本文提出一個方法能讓使用者取回文件後使用較少的運算即可解回原本的文件，這樣就能減輕手持行動裝置使用者的電力負荷，並且能夠保持文件安全。

關鍵詞: 虛擬亂數產生器，錯誤率，串流加密法，雜湊函數，搜尋加密文件。

Abstract

We want to store the documents on an untrusted server, and allow the documents to be searched by the server without leaking any information. We can encrypt the documents and create trapdoors for these corresponding documents. When we want to retrieve the documents, we just give the trapdoors to the server. An adversary can not get any information without the authorization of the user, even the server can know nothing from user's search request. In this paper, we propose a scheme which can retrieve the documents and decrypt more efficiently.

Keywords: pseudorandom generator, false positive rate, stream cipher, hash function, search encrypted data。

1 前言

近年來安全和隱私越來越被重視。現今的郵件系統如 IMAP 系統。人們必須對於儲存裝置的伺服器 (storage servers)— 例如檔案伺服器 (file servers)—有著完全的信賴。所以如果伺服器被敵人侵入或竊聽，那將會造成所存的文件將會變得不安全。但是如果不相信伺服器，不想洩漏任何資訊讓伺服器

知道, 則在取回文件時將會更加的困難, 因爲在伺服器上搜索文件將變得不容易實現。使用某些密碼學上的技術能使主機伺服器無法得到關於文件的任何資訊, 並且可以在加密的文件上創造一個暗門 (trapdoor) 使得當客户端想要搜索時能夠簡單的取得屬於自己的文件。在人們與組織公司越來越依賴線上伺服器去儲存大量資料的時代, 關於這一類型的搜索將會越來越被重視。如果某個使用者想要去存取在伺服器上的資料, 他們必須相信這並不會洩漏任何資訊給沒有被授權的人。這產生了一些架構在安全和隱私疑慮上的應用。

假設有一個攜帶個人數位助理 PDA (personal digital assistant) 的使用者叫做 Alice, 他希望能從一個不信任的郵件伺服器取回屬於自己的信件, 他可以先行做某些特定的加密, 使那些被加密的文件並不會被伺服器得到任何的資訊。當搜尋那些屬於 Alice 的文件時, Alice必須知道某些包含於文件中的關鍵字, 讓伺服器在執行搜尋和回應使用者之後不會洩漏關於文件的任何機密性。

本文提出一個在不洩漏任何機密的情況下能夠達成加密文件搜索某些關鍵字的方法。因爲人們時常透過手持行動裝置取得這些相關文件, 所以在此文中期望能在取得後經過簡單的運算解回原本的文件, 並且能夠保持相同的安全性。而在 Song 等人提出的方法 [1]中, 文件回傳給使用者後, 使用者解密的手續是非常繁複的。於是我們希望能夠在上傳文件時用多一點資訊使得當文件回傳時, 只需要一些簡單的步驟就能將密文解回明文, 並且能夠保持原來擁有的安全性。在此篇文章中將介紹一個方法能夠減少客户端取回文件後解密的手續。如此一來, 使用者能減少電池消耗量。

在第二部份我們將介紹加密文件搜尋欲達成的性質, 雜湊函數 (hash function) 的定義與串流加密法 (stream cipher) 的知識。第三部份則簡單敘述之前的方法與能改善的部份。接著介紹改進的方法於第四部份。第五部份將會分析搜索的碰撞率與安全性。最後, 在第六部份總結此方法的優缺點。

2 背景知識

此部份將敘述性質與關於所使用的函式之定義, 將會比較容易進行分析。在搜索加密文件時欲達成的性質在 [1],[2]中有提到如下:

1. 控制搜索 (Controlled searching): 伺服器沒有經過使用者的認證, 將無法任意進行任何有意義的搜索。
2. 隱藏搜尋 (Hidden queries): 伺服器將無法從使用者搜尋的關鍵字得到任何資訊。
3. 獨立詢問 (Query isolation): 伺服器無法從使用者搜尋結束後得到除了符合此次搜索的結果以外的額外資訊。
4. 獨立更新 (Update isolation): 在使用者更新文件時 (上傳新文件), 伺服器將無法得到任何額外的資訊。

一個假設爲均勻分佈 (uniform distributed) 的雜湊函數 (hash function) $f : A \rightarrow B$定義在 [3], [4]中有詳細介紹, 在此簡列如下:

1. 容易計算 (Easily computation): 對於任意一個x, 計算出$f(x)$是簡單容易的。
2. 固定位元長度 (Fix bit length): 一個雜湊函數f任意帶入一組二元數x當作輸入值, f將輸出一組固定長度的二元數$y = f(x)$當作他的輸出值。

3. 前映像抵禦 (Pre-image resistant): 給定任意一個輸出值$y = f(x)$, 計算出x使得$f(x) = y$幾乎是不可能實現的。

4. 碰撞抵禦 (Collision resistant): 給定一個x, 要找出另一個x'使得$f(x') = f(x)$幾乎是不可能計算出來的。

串流加密法 (Stream cipher): 在 [5]中有詳細介紹串流加密法, 本文在此簡述此法。串流加密法爲一種對稱式加密, 此方法爲將明文 (plaintext) 與虛擬亂數產生器 (pseudorandom generator) 所產生出的位元串結合, 得到密文C, 典型的方法爲透過 XOR 運算將兩數結合。此法安全性架構於虛擬亂數產生器不會被預測出下個亂數之位元。

3 相關研究

近年來, 個人隱私對於人們越來越重要。隨著關於搜索加密文件的技術越來越多 [2, 7, 8,12, 13, 14], 個人隱私問題也越來越被許多有興趣的人注意到。最先著眼到在加密的文件上做搜索的人是 Song 等人。他們展示出一個對稱式金鑰的方法允許使用者建立一個惟有自己知道的暗門。當使用者想要去查詢某些文件時, 可以只傳送先前加密的關鍵字給伺服器, 然後伺服器就能開始尋找相關於這個關鍵字的文件。但這方法仍然有一些小缺點值得去改進, 像搜尋的時間將會隨著文件的大小而增加, 這是因爲此法搜尋每份文件中每個加密的字。還有搜尋結束後回傳給使用者的文件必須經過多重手續才能解回原始文件。在剛才所提到關於伺服器搜尋時間的第一個問題是非常普遍的問題並且已經吸引了相當多的人做出改善方法。此篇文章著眼點在於第二個問題, 我們先簡敘 [1]中的方法, 接著再敘述此文所想要解決的問題。[1]中展示了四個方法—基本方法、控制搜索、隱藏搜尋、最終方法—都是架構在基本方法上。

首先, 介紹基本方法。使用者 Alice 擁有一個文件的集合$D = \{d_1, d_2, \ldots, d_l\}$共$l$份文件, 在 Alice 將這些文件上傳給伺服器之前, Alice 將會依文件把所有字$W_1, W_2, \ldots, W_t$分成同樣的長度n位元並排好, t爲一份文件中字數的總數量。爲了讓相同的字能產生不同的密文, 在加密每個字時所使用的是虛擬亂數產生器所產生的亂數$n - m$位元, 再將這些亂數用一個擬亂函數 (pseudorandom function) 與全相同或部份相異或全相異之金鑰k_i產生後面m位元, 此處i爲相對應於每個字之位置。接著將明文與此亂數做 XOR 運算, 如此一來相同的字就不會產生同樣的密文。亂數、金鑰被寫爲$T_i = < S_i || F_{k_i}(S_i) >$此形式, 此處之$||$爲串聯的意思。接著將這些字做XOR 運算使得$C_i = W_i \bigoplus T_i$, 得到密文$C_1, C_2, ..., C_t$, 此程序將作用於$D$中的每份文件, 然後將$D$上傳至伺服器, 加密方法如圖 1。

接著敘述此方法如何進行搜尋與解密。首先, Alice 將希望搜索的字W與對應於此W的金鑰k一併傳給伺服器, 伺服器收到後將每份文件中的每個密文都與W做 XOR 之運算後, 用擬亂函數與收到之金鑰k測試, 輸入前$n - m$位元 (L_i) 是否與後m位元 (R_i) 相符合, 亦即, 測試$F_k(L_i) = R_i$。如果測試結果相同則將此文件傳回給Alice, 若整份文件中所有密文皆不符合, 則判斷此份文件並不是 Alice 所要的文件。搜尋方法如圖 2所示。

由於此方法不符合欲達成的性質, 於是 Song 等人將此架構修改小部份使得之後的方法能符合欲達成之性質, 此文中將直接介紹 Song 等人最後得到的方法。在 Alice 建構時, 將每份文件的$W_1, W_2, \ldots, W_t$先行加

密成$E_k(W_i)$，並將$E_k(W_i)$分爲$< L_i||R_i >$，L_i爲$E_k(W_i)$之前$n-m$位元，R_i爲後m位元，此處k爲使用者建立之私密金鑰，$i = 1,2,\ldots,t$。接下來將$E_k(W_i)$與T_i做XOR 運算，使得$C_i = E_k(W_i) \bigoplus T_i$，此處$T_i =< S_i||F_{k_i}(S_i) >$，$S_i$爲虛擬亂數產生器產生之長度爲$n-m$的亂數，$F$爲擬亂函數，$k_i = f_{k'}(L_i)$，$f$與$F$爲相異之擬亂函數，$k'$爲使用者建立之私密金鑰，與$k$相異，即$k \neq k'$。接著 Alice 將每份文件上傳至伺服器，搜尋關鍵字W時則將$E_k(W)$與$k = f_{k'}(L)$傳給伺服器，此處L爲$E_k(W)$之前$n-m$位元，建構與搜尋方法如圖3。

而在 Goh[6]的安全索引 (Secure Indexes) 中，所使用的方法爲利用 bloom 過濾器 (Bloom filter[9])，對每份文件d_i中之關鍵文字$S = \{s_1, s_2, \ldots, s_n\}$放入bloom 過濾器中，此處關鍵文字由使用者決定。此方法詳述如下，由使用者選定r個雜湊函數 (hash function)$h_1, \ldots, h_r$，此處$h_i : \{0,1\}^* \rightarrow [1,m]$，$m$之大小決定於關鍵文字之數量$n$與雜湊函數之個數$r$。將$S$中每個文字都放入雜湊函數中，如$h_1(s_i), h_2(s_i), \ldots, h_r(s_i)$，接著若$h_j(s_i) = v$則將長度爲$m$之陣列的第$v$格設定爲1，若有碰撞則不予理會，其餘皆設定爲0。此方法雖能在常數時間內完成使用者欲搜尋之文字，但仍然有搜尋錯誤及無法搜尋片語或成語之缺陷。

4 改進方法

在這部份介紹取回文件後解密手續複雜的解法。首先，在建構階段 Alice 擁有一個文件的集合$D = \{d_1, d_2, \ldots, d_l\}$共$l$份文件，Alice將每份文件的字拆成$n$位元相同長度的字$W_1, W_2, \ldots, W_t$，其中$t$爲每份文件中字數的總數量。將每個文字用串流加密法 (stream cipher) 如 RC4[10]，加密成爲密文$C_{11}, C_{21}, \ldots, C_{t1}$。再將每個文字使用對稱式加密法 (symmetric-key crypto-systems) 如 AES[11]加密過後得到$E_k(W_i)$，爲了讓相同的文字能得到不同的密文，我們將$E_k(W_i)$串聯其文章編號與其在文章中的字數成爲$< E_k(W_i)||j||i >$，並將此值輸入雜湊函數得到$C_{i2} = h(< E_k(W_i)||j||i >)$，此處$j = 1,2,\ldots,l$, $i = 1,2,\ldots,t$。然後把每份文件中的C_{i1}與C_{i2}上傳至伺服器，建構方法如圖4。

搜尋階段，Alice 將其所欲搜尋之字W加密過後得到$E_k(W)$傳給伺服器，伺服器對每份文件的每個C_{i2}與$h(< E_k(W)||j||i >)$做比對，此處$j = 1,2,\ldots,l$, $i = 1,2,\ldots,t$，若有相同之值則將$C_{11}, C_{21}, \ldots, C_{t1}$回傳給 Alice。Alice 收到$C_{11}, C_{21}, \ldots, C_{t1}$後，直接用串流加密法之金鑰解回原本的明文$W_1, W_2, \ldots, W_t$。

5 分析

在此部份將分析此文的方法與之前方法的差異，以及搜尋文件的錯誤率。首先，我們的方法將 Song 等人的方法 [1]改進，讓使用者取回文件後僅需一個步驟便能解回文件。在 [1]中，當使用者傳回文件時，必須進行六個步驟，

Step 1. 使用虛擬亂數產生器與亂數種子 (seed) 產生長度爲$n-m$位元之亂數S_i。

Step 2. 將X_i與S_i做XOR 運算得到$L_i = X_i \bigoplus S_i$，此處X_i爲取回之密文C_i的前$n-m$位元。

Step 3. 接著利用L_i與金鑰k'帶入擬亂函數f得到W_i之金鑰$k_i = f_{k'}(L_i)$。

Step 4. 再用亂數S_i與金鑰k_i帶入擬亂函數F得到長度爲m位元之$F_{k_i}(S_i)$。

Step 5. 將取回之密文C_i與$< S_i||F_{k_i}(S_i) >$做XOR 運算, 得到$E_k(W_i) = C_i \bigoplus < S_i||F_{k_i}(S_i) >$。

Step 6. 最後再將$E_k(W_i)$解密回W_i。

而在我們的方法中,取回文件後僅須做一個步驟, 即可解密回明文, 則能節省相當多運算。並且在客户端所需保存的資訊也減少了, 在[1]中, 客户端須保留的資訊有虛擬亂數產生器、兩個擬亂函數、亂數種子、擬亂函數用的金鑰k'與加密文字用的金鑰k, 然而本文方法僅須記住搜尋時加密文字用的金鑰k, 虛擬亂數產生器與串流加密所使用的亂數種子。

在 Song 等人的方法 [1]與本文方法中, 皆有可能搜尋到錯誤的文件, 換句話說, 有可能D_j中沒有W這個字, 但搜尋時卻傳回D_j。會造成這樣的情況主要是因爲兩方法皆使用XOR 運算, 所以即使兩個不相同的字也有可能得到相同的結果, 在這裡簡單估算搜尋到錯誤文件的機率。給予一個欲搜尋的文字$E_k(W)$, 雜湊函數 (hash function) 爲均勻分佈, 且雜湊函數產生之值長度取前p個位元, 若文件中有t個字, 則搜尋此文件之錯誤率 (false positive rate) 爲$fp = t/2^p$(若此文件不含W這個字)。所有文件之平均錯誤率的算法爲 avg $fp = l * t/2^p$, 其中l爲總文件數, t爲文件平均字數, fp爲錯誤率。假設一份文件有10^6個字, 雜湊函數取32位元, 則此份文件之錯誤率爲$fp = 10^6/2^{32} \approx 1/4000$, 若有 1000 份不含$W$的文件, 平均每份文件有$10^6$個字, 則搜尋到錯誤文件的機率爲$1/4$。若希望平均錯誤率爲$1/1000$則雜湊函數約需取$p \approx \lg \frac{t*l}{1000}$位元, 此處$l$爲總文件數, t爲文件平均字數。第二種方法爲允許每份文件取相異位元之雜湊值, 換句話說, 若D_1有10^6個字, D_2只有10^3個字, 期望兩份文件錯誤率皆爲$fp = 1/1000$, 則D_1之雜湊值可取30個位元, 而D_2可取20 個位元。如果使用後述之法將可減少小文件上傳至伺服器時所佔之空間。

在這裡可以簡單的看出此文的方法中確實有做到第二部份所提到之性質, 在控制搜索 (Controlled searching) 部份, 給予伺服器之值$E_k(W)$伺服器僅能搜尋到正確的字, 無法做任意的搜尋。而且給伺服器的$E_k(W)$亦不會洩漏任何關於文件的資訊, 這滿足了隱藏搜尋 (Hidden queries)。獨立詢問 (Query isolation) 部份, 由於搜尋方面與文件內容的部份無關所以伺服器亦無法從搜尋的結果得到任何關於文件的資訊。最後, 關於獨立更新 (Update isolation), 當上傳一份新的文件時, 作法完全與先前的文件獨立, 所以並不會透露關於任何先前的文件的資訊。最後, 我們在表1比較三種方法。

6 結論

這篇文章敘述了一個在建構時多使用一些空間而能讓使用者在解密時能花費較少手續的方法, 在我們的方法中, 能夠搜尋連續兩個字以上的片語或成語。而且如果伺服器收到C_{i2}時能將之排序, 那伺服器搜尋時將能花費更少的時間, 由$O(n)$降低至$O(\lg n)$。在本篇文章中能達到先前文章的安全性, 並且改進了之前文章的缺點—解密的手續與使用者必須儲存的多餘資訊, 而且此文使用的加密法相當簡易, 所使用的串流加密法與雜湊函數都是相當快速的加密法, 對於使用者的負擔能有相當程度的降低。

致謝

本研究由國科會補助。

計畫編號: NSC96-2628-E-005-076-MY3。

7 參考文獻

[1] D. Song, D. Wagner, and A. perrig, *Practical Techniques for Searches on Encrypted Data*, in Proc. of the 2000 IEEE symposium on Security and Privacy (S&P 2000).

[2] S. Artzi, A. Kiezun, C. Newport, D. Schultz, *Encrypted Keyword Search in a Distributed Storage System*, in DSpace at MIT, 2006.

[3] M. Naor, M. Yung, *Universal One-Way Hash Functions and their Cryptographic Applications*, in Proc. of the twenty-first annual ACM symposium on Theory of computing, 1989.

[4] Jenkins, Bob, *Hash Functions*, Algorithm Alley, Dr. Dobb's Journal, 1997.

[5] Matt J. B. Robshaw, *Stream Ciphers Technical Report TR-701*, version 2.0, RSA Laboratories, 1995.

[6] Eu-Jin Goh, *Secure Indexes*, Technical Report 2003/216, IACR ePrint Cryptography Archive, 2003.

[7] D. Boneh, G. di Crescenzo, R.Ostrovsky, and G. Persiano, *Public key encryption with keyword search*, in Proc. Eurocrypt 04, pages 506-522, 2004.

[8] Y. C. Chang and M. Mitzenmacher, *Privacy preserving keyword searches on remote encrypted data*, in Applied Cryptography and Network Security Conference (ACNS), 2005.

[9] B. Bloom, *Space/time trade-offs in hash coding with allowable errors*, in Communications of the ACM, 1970.

[10] Ilya Mironov, *Random Shuffles of RC4*, CRYPTO pages 304-319, 2002.

[11] J. Daemen, V. Rijmen, *AES Proposal: Rijndael*, 1999.

[12] R. Curtmola, J.Garay, S.Kamara, and R. Ostrovsky, *Searchable symmetric encryption: Improved definitions and efficient constructions*, Cryptology ePrint archive, 2006.

[13] R. Brinkman, J. Doumen, W. Jonker, *Using secret sharing for searching in encrypted data*, Secure Data Management VLDB 2004 workshop, 2004.

[14] R. Brinkman, L. Feng, S. Etalle, P. Hartel, and W. Jonker, *Experimenting with linear search in encrypted data*, Centre for Telematics and Information Technology, 2003.

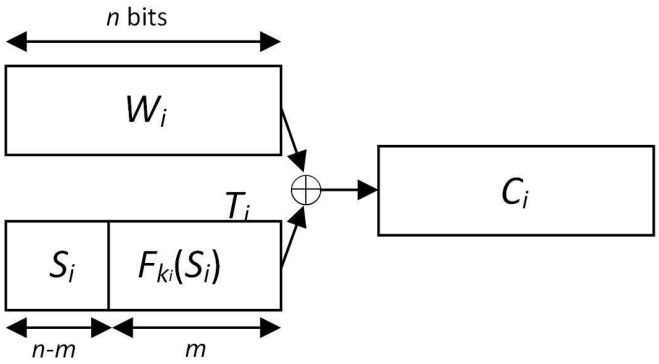

圖 1：基本方法

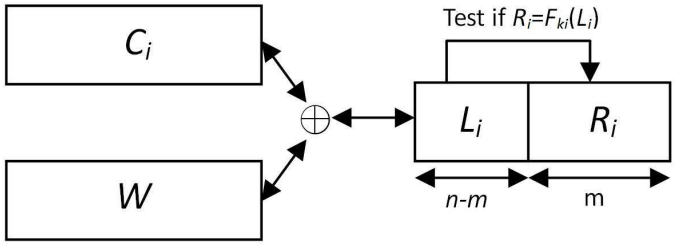

圖 2：基本方法搜索關鍵字

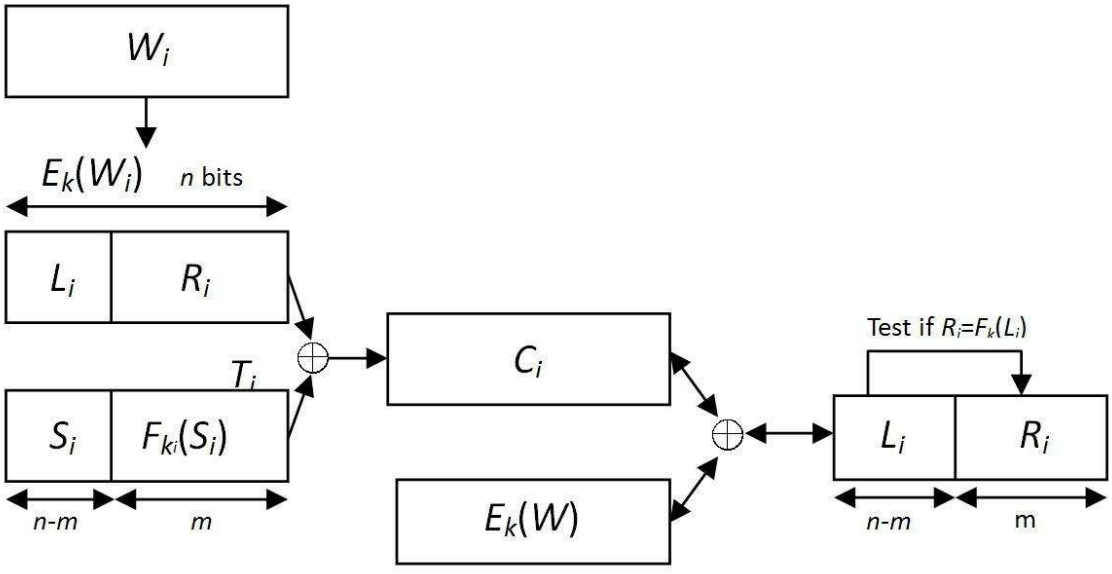

圖 3：最終方法建構與搜索關鍵字

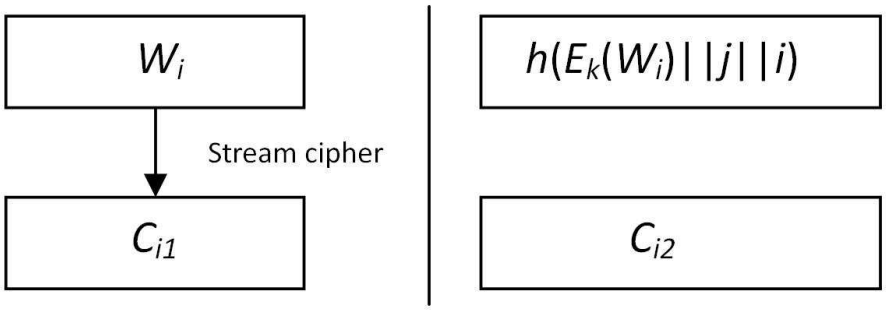

圖 4：改善 Song 等人方法之建構

表 1：各方法比較

	本文方法	Goh[6]	SWP[1]
伺服器搜尋時間	$l*O(t)$	$l*O(1)$	$l*O(t)$
使用者解密步驟	1 step (視文件加密法)	1 step (視文件加密法)	6 steps
伺服器空間使用	$2l$	$l+l*n*r$	l
片語與成語搜尋	能	否	能

其中 l 爲文件數, t 爲各文件字數, n 爲使用者挑選之關鍵字數.