用於搜尋加密文件的索引架構

林宜正
國立中興大學
資訊科學與工程學系研究生
s9556022@cs.nchu.edu.tw

洪國寶
國立中興大學
資訊科學與工程學系教授
gbhorng@cs.nchu.edu.tw

摘要

索引(Index)搜尋為一種加速文件關鍵字搜尋的一種架構，但是隨著文件的安全性越來越受到重視，文件開始以加密的型態儲存於資料庫，但是目前的索引架構並沒有針對加密文件作考量，也因此索引有可能會透露出文件的相關資訊，造成安全上的問題。本文提出了為加密文件設計的索引搜尋架構(PRP-Index)，即使加密的文件資料及索引儲存於不被信任的伺服器端(Server)上，也不會有安全上的疑慮，並且達到零錯誤率的搜尋以及對片語(Key Phrase)搜尋的支持，同時保有索引搜尋的效率。另一個優點是此索引架構不會影響文件的加密方式，讓使用者可以依據各種需求來使用不同的加密技術，使加密更有彈性。

關鍵詞：索引、雜湊函式、Pseudorandom Permutation、Pseudorandom Function、加密搜尋

Abstract

Keyword index is a data structure that allows us to search in constant time for documents containing specified keywords. Unfortunately, standard index constructions are not designed for encrypted documents, because they may leak some information about the document. In this paper, we propose an index structure called PRP-Index which is designed for encrypted documents. Even if these encrypted documents and their indexes are stored on an untrusted server, there will no secure problems. And PRP-Index can achieve certain search and support key phrase search. Another advantage is that the index structure and the document's encryption techniques are independent. User can choose any encryption technique they want to use for their different kind of requirements.

Keywords: Index, Hash Function, Pseudorandom Permutation, Pseudorandom Function, Encrypted Keyword Search.

1. 前言

現在已經進入了數位的時代，越來越多的資料都儲存在電腦之中，很多私人或機密性資料也儲存於伺服器端的資料庫中，並藉由網路來進行存取，因此維護這些資料的安全，在近年來也越來越受到重視[6]。維護安全性的方法最主要的方式就是透過加密，但是這也衍生出其它問題，因為在進行搜尋通常是用索引(Index)來加強搜尋效率，但是目前的索引搜尋都是為明文搜尋所設計，對加密過的資料並不適用，而且若資料和索引儲存在不信任伺服器端(Server)上，那麼索引會不會透露機密資料的資訊的可能性也需要納入考量。

另一種方法是將所有加密文件直接傳給使用者端(Client)而不經過搜尋，但是當文件數增加時會相當浪費網路的頻寬，尤其是當使用者端是使用資源及運算能力都相當有限的行動裝置(Mobile Devices)時，例如：手機、個人數位助理(Personal Digital Assistant)等，便更加不適用了，因此加密文件的搜尋是必要的。本文建立了為加密資料所設計的索引搜尋架構，為搜尋的關鍵字建立所謂的後門(Trapdoor) ，藉由關鍵字的後門(Trapdoor)和索引，伺服器端便能在短時間內有效的搜尋到使用者所要搜尋的資料，而且索引和關鍵字的後門不會洩漏任何有關加密資料的訊息。

Song 等人[4]為加密文件的搜尋定義了三個需要遵守的特性：(1)伺服器端無法從搜尋結果得到更多明文的訊息(Query Isolation) (2)伺服器端在沒有使用者的認證下無法進行搜尋(Controlled Searching) (3)使用者欲搜尋的關鍵字並不會直接傳給伺服器端，而是傳關鍵字的後門(Trapdoor)，如此伺服器端無法得知欲搜尋的關鍵字是什麼，但是還是能進行搜尋(Hidden Queries)。這三個特性主要是維護加密資料儲存

在不信任第三方的資料庫上的安全性，讓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的資訊。而本文所提出的索引搜尋架構也是建立在將資料儲存於不信任的第三方伺服器端上，因此也需滿足以上所要求的三個特性。

本文第二部分將描述對於加密文件搜尋的相關研究及問題，第三部分會先講解我們提出的索引方法所用的技術為何，再講解其架構，並在 3.3 會和 Goh 的 Bloom filter Index 做個比較，分析其搜尋時間、花費空間及安全度。

2. 相關研究

目前有關於加密文件的搜尋技術有許多種，例如：[3, 7]使用公開金鑰的技術，使用者端利用私鑰為關鍵字建立後門，伺服器端使用公鑰來進行搜尋。[14]使用對稱式金鑰加密來建立主索引，只需搜尋此索引即可指出所有文件，但更新索引所需成本很高。也有使用樹狀結構來加強搜尋之效率[12]。而[2]使用秘密分享的方式來專為 XML 進行搜尋以及其他方法[8, 9, 10, 13, 15]等等。

Goh[5]提出了利用 Bloom Filter[1]的方法來建構索引(不同的文件對應到不同的索引)，其中 Bloom Filter 是一種搜尋過濾器，它使用的核心技術為雜湊編碼(Hash Coding)，其特點為利用較少空間存放關鍵字集合，再利用雜湊編碼進行計算來做搜尋，例如:有一個文件 D 及它所包含的文字的集合 $S = \{s_1, s_2, \ldots, s_n\}$，$n$ 表示由使用者所選的關鍵字個數，這些關鍵字的集合將由大小為 m bit 的陣列所表示，藉由將集合裡的關鍵字代入 r 個獨立的雜湊函式(Hash Function) h_1~h_r 中，其中對於 $i = 1$ 到 r，$h_i : \{0, 1\}^* \rightarrow [1, m]$，將 S 中的每一個關鍵字放入雜湊函式中得到 $h_1(s_i), \ldots, h_r(s_i)$，將這 $n * r$ 個所對應到陣列位址的值設為 1，便完成將 S 放入陣列的設定。

若要搜尋關鍵字 a 是否屬於 S，則要將搜尋的關鍵字 a 代入 r 個雜湊函式，再到陣列所對應位址查詢值是否皆為 1，若都是 1 表示 $a \in S$，否則 $a \notin S$。由於是使用雜湊函式，搜尋的時間平均在常數時間(Constant Time)內便能完成搜尋，相當有效率。但是雜湊函式會發生碰撞(Collision)的問題，而產生誤判(False Positive)的情況，例如：有一個關鍵字 $s_i \notin S$，但是在經由 Bloom Filter 進行判斷後卻都得到 $s_i \in S$，便產生了誤判，如圖 1 所示。

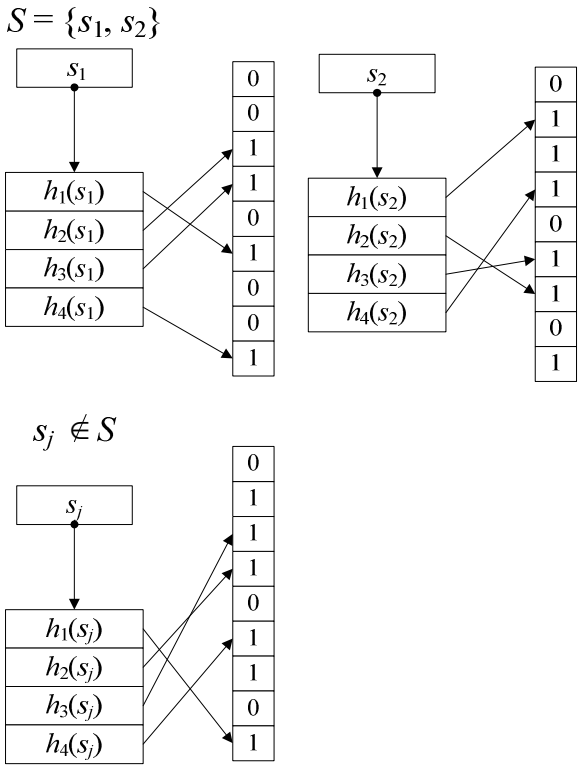

圖 1. Goh Bloom Filter

除了不能 100%正確的判斷外，Bloom Filter 僅能搜尋單字，對於片語(Key Phrase)也無法進行搜尋，但是大部分的使用者搜尋時不只是搜尋單字亦也搜尋片語來加強要搜尋的文件的準確率，但 Bloom Filter 僅能判斷片語裡的字是否在文件裡，而不能分辨其順序。而本文所提出的索引架構也將這方面考量進去，以達到 100%的準確判斷和對片語搜尋的支援並維持效率。

3. 索引架構

3.1 背景資料

Pseudorandom Function(PRF)

Pseudorandom Function 為產生偽隨機數的函式，單向且隨機，例如：給予$(x_1, f(x_1, k)), \ldots, (x_m, f(x_m, k))$，則惡意攻擊者對於任意 x_{m+1} 無法預測其 $f(x_{m+1}, k)$為多少。若我們稱這個函數 f: $\{0, 1\}^n \times \{0, 1\}^s \rightarrow \{0, 1\}^m$ 為 (t, ε, q)-pseudorandom-function，則此函式滿足下列性質：

1. $f(x, k)$可以定義為 $f_k(x)$，計算速度快，其中輸入值 $x \in (0, 1)^n$ 以及金鑰 $k \in \{0, 1\}^s$。
2. 在 t 的時間，裡演算法 A 進行了 q 次查訊後：

$|\Pr[A^{f(.,k)} = 0 \mid k \leftarrow \{0,1\}^s] - \Pr[A^g = 0 \mid g \xleftarrow{R} \{F: \{0,1\}^n \rightarrow \{0,1\}^m] | < \varepsilon$, 。

Pseudorandom Permutation(PRP)

同樣也是隨機，不過是將輸入值的 bit 做隨機的排序，也就是輸出值的 0, 1 的個數和輸入值相同，但位置不同。在相同金鑰下，不會產生輸入值不同，而輸出值相同的情況，不會產生碰撞(Collision)，而且為單向函式。

若我們稱這個函數$\prod$: $\{0,1\}^n \times \{0,1\}^s \rightarrow \{0,1\}^n$ 為(t, ε, q)-pseudorandom-permutation，則此函式滿足下列性質：

1. $\prod (x, k)$可以定義為 $\prod_k(x)$，計算速度快，其中輸入值 $x \in (0,1)^n$ 以及金鑰 $k \in \{0,1\}^s$。
2. 在 t 的時間裡，演算法 A 進行了 q 次查訊後：

$|\Pr[A^{\prod, \prod^{-1}} = 0] - \Pr[A^{E, E^{-1}} = 0]| < \varepsilon$, 其中 E 爲 random permutation。

3.2 架構分析

我們提出了二種索引架構，稱為 PRP-Index1 及 PRP-Index2，由 3 個函式組成，分別為 BuildIndex(D, D_{id})、SearchIndex(T_w, I_d) 及 Trapdoor(w, k)，其中 D_{id} 為文件 D 的編號，$W = (w_1, \ldots, w_n)$為文件 D 中的關鍵字，f 為 pseudorandom function，$\prod$ 為 pseudorandom permutation，k 為使用者持有的金鑰。

PRP-Index1

- **設定階段**

BuildIndex(D, D_{id}): 為文件 D 建立索引

1. 首先將使用者所選擇的關鍵字集 W 裡每一個關鍵字加密得到其後門(trapdoor)為 $E_k(w_1), \ldots, E_k(w_n)$ 。
2. 之後以文件的編碼 D_{id}(視為 id)為金鑰，將每個關鍵字的後門代入 pseudo-random function，得到 codeword $x_1 = \prod_{id}(E_k(w_1)), \ldots, x_n = \prod_{id}(E_k(w_n))$。
3. 最後將 $x_1, \ldots, x_n$ 經過排序後放入陣列 d 中，陣列裡每一個格存放二個值，其中一個為存放關鍵字的 codeword，另一個儲存指標，指向一個大小為 m 的陣列 s，其中 m 為文件 D 所有字的個數，指標會根據關鍵字所在的位置來指向陣列 s。例如：假設關鍵字 w_4 出現在文件 D 的位置為第 2, 4 個字，則陣列 d 裡的指標會指向 s[2]，而 s[2] 裡儲存另一個指標指向 s[4]，s[4]存的指標為 null，陣列 s 負責指出關鍵字的位置，同一個關鍵字 w_i 的指標由 $f_{id}(E_k(w_i))$當作金鑰進行加密，所以上面 3 個指標皆用 $f_{id}(E_k(w_4))$作為金鑰來加密。陣列 d 的指標為開頭如同鏈結串列，如圖 2 所示。

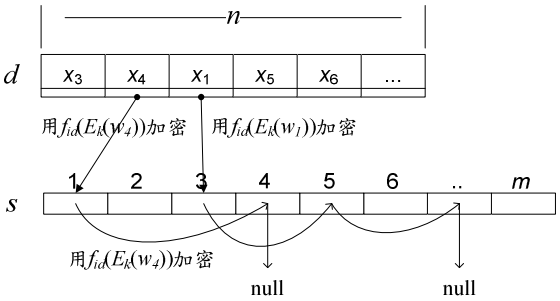

圖 2. PRP-Index1

W 裡的關鍵字經過以上的運算放入陣列 d, s 後，便完成文件 D 的索引設定，BuildIndex(D, D_{id})會回傳索引 $I_{id} = (d, s)$。

- **搜尋階段**

使用者將加密文件及其索引和文件編號傳給伺服端，之後便可以進行搜尋。

1. 使用者端執行 Trapdoor(w, k)產生要搜尋的關鍵字 w 的後門(trapdoor) $T_w = E_k(w)$，將 T_w 傳給伺服器端。
2. 伺服器端在接收到 T_w 之後，開始對資料庫裡每一個文件的索引進行搜尋 SearchIndex(T_w, I_{id})，將文件的編號 D_{id} 作為金鑰(視為 id)，對 T_w 進行計算得到 codeword $x = \prod_{id}(E_k(w_i))$。
3. 對文件 D 索引裡的陣列 d 作二元搜尋(Binary Search)，若沒有搜尋到，回傳 0，若有則回傳 1。其中若使用者是查詢片語，則在搜尋到之後計算 $f_{id}(E_k(w))$來將指向陣列 s 的指標解密，依據指標到陣列 s 裡搜尋片語中關鍵字的位置，最後根據關鍵字的相對位置來判斷搜尋的片語是否儲存於此文件 D 中，如圖 3。

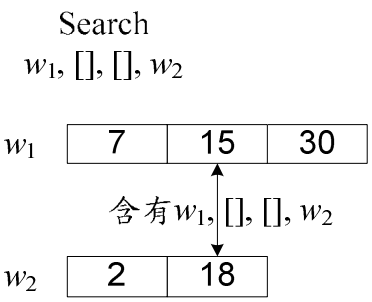

圖 3. Search Key Phrase

- **特性**
  1. 由於 pseudo-random permutation 在同一金鑰不會碰撞的特點下，同一文件裡的關鍵字的 codeword 不會重覆，所以伺服器端能 100%準確地搜尋。
  2. 在搜尋時因為已排序完成，所以用二元搜尋時，其搜尋時間為 O(log n) 。
  3. 使用文件的編碼作為此文件索引的金鑰，因此對於不同文件無法利用索引來查出其相關性，例如：是否有相同的關鍵字等等。
  4. 增加一些空間來支援對片語的搜尋，由於索引和文件互相獨立的，所以文件的加密不會因索引而受限，因此使用者可以更根據需求來使用不同的加密法，例如：對於行動裝置，可以使用解密速度較快、較省資源的加密法，讓加密相當有彈性。

PRP-Index2

- **設定階段**

  1, 2 和 PRP-Index1 相同。

  3. 將關鍵字的 codeword 代入一個雜湊函式 h:{0, 1}* → [1, n]，得到介於 1 到 n 的值，雜湊函式 f 根據文件的關鍵字個數調整 n ，使得 n 為關鍵字的個數，因此不同的文件使用不同的雜湊函式 f。將 codeword 放入一陣列 d，其大小為 n，根據雜湊函式所計算出來的值放到陣列 d 所對應的位置，若有碰撞先以鏈結串列表示，如圖 4 所示。在插入所有關鍵字的 codeword 之後，將碰撞產生的鏈結串列的值放入陣列 d 裡剩餘的位置並用指標記錄起來，如圖 5 所示。

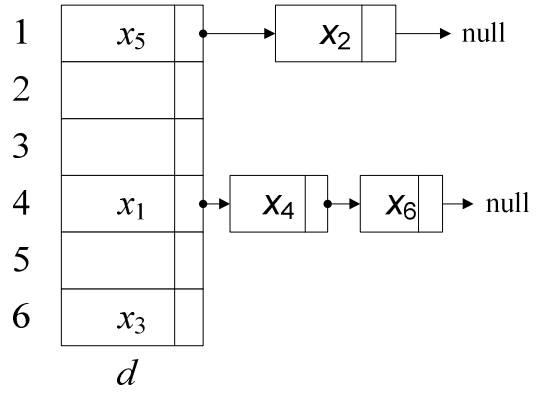

圖 4. 碰撞處理

陣列 d 除了儲存 codeword、碰撞指標以外，還有儲存另一指標指向一個大小為 m 的陣列 s，m 為文件 D 所有字的個數，指標會根據關鍵字所在的位置來指向陣列 s。例如: 假設關鍵字 w_5 出現在文件 D 的位置為第 2, 7 個字，則陣列 d 裡的指標會指向 s[2]，而 s[2]裡儲存另一個指標指向 s[7]，s[7]存的指標為 null，陣列 s 負責指出關鍵字的位置，同一個關鍵字 w_i 的指標由 $f_{id}(E_k(w_i))$當作金鑰進行加密(注意:碰撞指標沒有加密)，所以上面 3 個指標皆用 $f_{id}(E_k(w_5))$作為金鑰來加密。陣列 d 的指標為開頭如同鏈結串列，如圖 5 所示。將 W 裡的關鍵字經過以上的運算放入陣列 d, s 後，便完成文件 D 的索引設定，BuildIndex(D, D_{id})會回傳索引 I_{id} = (d, s)。

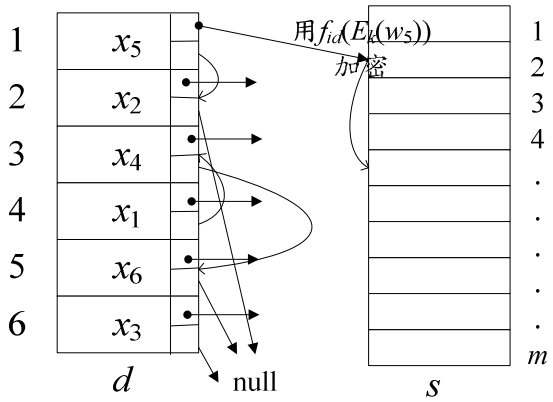

圖 5. PRP-Index2

- **搜尋階段**

使用者將加密文件及其索引和文件編號傳給伺服端，之後便可以進行搜尋。

1. 使用者端執行 Trapdoor(w, k)產生要搜尋的關鍵字 w 的後門(trapdoor) $T_w = E_k(w)$，將 T_w 傳給伺服器端。
2. 伺服器端在接收到 T_w 之後，開始對資料庫裡每一個文件的索引進行搜尋 SearchIndex(T_w, I_{id})，將文件的編號 D_{id} 作為金鑰(視為 id)，對 T_w 進行計算得到 codeword $x = \prod_{id}(E_k(w))$。
3. 將 codeword 代入雜湊函式 h，根據得到的值到陣列 d 中去尋找，若有找到則回傳 1，若沒有，則看鏈結串列的指標指向 d 的其他位置是否存有欲搜尋的關鍵字，搜尋直到指標指向 null，若都沒有搜尋到則回傳 0。其中若使用者是查詢片語，則在搜尋到之後計算 $f_{id}(E_k(w))$來對指向陣列 s 的指標解密，依據指標到陣列 s 裡搜尋片語中關鍵字的位置，最後根據關鍵字的相對位置來判斷搜尋的片語是否儲存於此文件 D 中，如圖 3。

- **特性**
  1. 由於 pseudorandom permutation 在同一金鑰不會碰撞的特點下，同一文件裡的關鍵字的 codeword 不會重覆，所以伺服器端能 100%準確地搜尋。
  2. 因為使用雜湊函式，即使有碰撞的情形，搜尋時間平均仍為 O(1)[11]。
  3. 使用文件的編碼作為此文件索引的金鑰，因此對於不同文件無法利用索引查出其相關性，例如：是否有相同的關鍵字等等。
  4. 增加一些空間來支援對片語的搜尋，由於索引和文件互相獨立的，所以文件的加密不會因索引而受限，因此使用者可以更根據需求來使用不同的加密法，例如：對於行動裝置，可以使用解密速度較快、較省資源的加密法，讓加密相當有彈性。

3.3 效能比較

在這裡將 Goh Bloom Filter 的索引和我們提出的 PRP-Index 做比較。

n 為文件中使用者所挑選的關鍵字個數，m 為字的長度，r 為 Bloom filter 雜湊函式的個數。

在更新(Updata)索引方面，分為三項–插入、刪除和修改。Goh 的方法因為有碰撞問題，因此必須重新設定整個索引來做更新，所花費的時間為 O(n)。 PRP-Index1 使用排序方法儲存於陣列之中，因此更新所花的時間跟陣列長度有關為 O(n)。PRP-Index2 藉由雜湊函式能迅速地找需要修改的關鍵字，即使碰撞也有碰撞機制使其能正確地找到需要更新的關鍵字，因此更新時間和搜尋時間相同為 O(1)。

表 1 三個索引的比較

	Goh Bloom Filter	PRP-Index1	PRP-Index2
Exact Location	×	√	√
Key Phrase	×	√	√
Controlled search	√	√	√
Query Isolation	√	√	√
Hidden Queries	√	√	√
Data Type	Any	Any	Any
Search Time (per doc)	O(1)	O(log n)	O(1)
Space Cost (per doc)	O(nr)	O(nm)	O(nm)
Encryption Method	Any	Any	Any
Update Time (per doc)	O(n)	O(n)	O(1)

由表 1 可以看出 PRP-Index2 和 Goh 的 Bloom Filter Index 含有較佳的搜尋時間 O(1)，但僅 PRP-Index1 和 PRP-Index2 具備零錯誤率的搜尋及對片語的搜尋的支持；在更新索引方面，PRP-Index2 所花的時間較少。

4. 結論

本文提出了兩種建構在加密文件搜尋的索引架構，最主要的特色為具有較彈性的加密、零錯誤率的搜尋以及對片語搜尋的支持，同時仍保有一定的搜尋效率，這些特點幫助使用行動裝置的使用者節省許多的頻寬及運算資浪費。因為目前是以空間換取時間的作法加強索引的能力，所以未來希望能在維持以上特點的前提下，將伺服器端空間的消耗降到最低，並結合其它搜尋技術，以提升加密索引搜尋的整體效能。

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] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," ***In Communications of the ACM***, pp.422-426, July 1970.

[2] Brinkman, R. and Schoenmakers, B. and Doumen, J.M. and Jonker, W, "Experiments with Queries over Encrypted Data Using Secret Sharing," ***In Secure Data Management (SDM)***, pp.33-46, 2005.

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

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

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

[6] G. Amanatidis, A. Boldyreva, and A. O'Neill, "New security models and provably-secure schemes for basic query support in outsourced databases," ***In Working Conference on Data and Applications Security (DBSec '07)***, 2007.

[7] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," ***In Cryptology ePrint Archive***, 2005.

[8] L.T.A. Joseph, A. Samsudin and B. Belaton, "Efficient Search on Encrypted Data", ***In Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication***, pp.6, Nov 2005.

[9] M. Bellare, A. Boldyreva and A. O'Neill, "Deterministic and Efficiently Searchable Encryption", ***In Advances in Cryptology - CRYPTO 2007***, pp.535-552, Aug 2007.

[10] M. Freedman, Y. Ishai, B. Pinkas and O. Reingold, "Keyword Search and Oblivious Pseudorandom Functions," ***In Proceedings of 2nd Theory of Cryptography Conference (TCC '05) Cambridge***, Feb 2005.

[11] R. Brinkman, L. Feng, J. M. Doumen, P. H. Hartel, and W. Jonker, "Efficient tree search in encrypted data," ***In Technical Report TR-CTIT-04-15***, Mar 2004.

[12] R. Brinkman, L. Feng, S. Etalle, P.H. Hartel and W. Jonker, "Experimenting with linear search in encrypted data," ***In Center for Telematics and Information Technology, University of Twente***, Dec 2005.

[13] R. Curtmola, J. Garay, S. Kamara and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," ***In ACM conference on Computer and communications security***, pp.79-88 ACM CCS 2006.

[14] S. Bellovin and W. Cheswick, "Privacy-enhanced searches using encrypted bloom filters," ***In Cryptology ePrint Archive***, 2004.

[15] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein, ***Introduction to Algorithms***, 2th ed., MIT Press and McGraw-Hill, pp.221-252, 2001.