SEMINAR REPORT
ON
PRIVACY PRESERVING SIMILARITY BASED TEXT RETRIEVAL

SUBMITTED BY
SONY P
M.TECH CIS
2010-2012 BATCH
ABSTRACT

Users of online services are increasingly wary that their activities could disclose confidential information on their business or personal activities. It would be desirable for an online document service to perform text retrieval for users, while protecting the privacy of their activities. A new similarity-based text retrieval scheme that prevents the server from accurately reconstructing the term composition of queries and documents, and anonymizes the search results from unauthorized observers. At the same time, this scheme preserves the relevance-ranking of the search server, and enables accounting of the number of documents that each user opens. The effectiveness of the scheme is verified empirically with two real text corpora.

Keywords

General Terms: Security

Additional Key Words and Phrases: Privacy of search queries, security in text retrieval, singular value decomposition


**Privacy preserving similarity based text retrieval**

**Table of Contents**

1. INTRODUCTION ................................................................. 5
2. BACKGROUND ON TEXT RETRIEVAL ......................................... 7
   2.1 Keyword Matching .......................................................... 7
   2.2 Similarity-Based Retrieval ............................................... 7
3. DOCUMENT RETRIEVAL MODEL ................................................. 9
4. SOLUTION APPROACH ............................................................ 11
5. DETAILED INDEXING AND RETRIEVAL ........................................ 13
   5.1 CORPUS PREPARATION .................................................... 13
   5.2 Query Processing .......................................................... 13
   5.3 Retrieval of Result Documents ......................................... 15
6 PROTOCOL ANALYSIS OF PRIVACY SAFEGUARDS ......................... 16
   6.1 Scenario A: Independent Document Server and Access Manager .......... 16
   6.2 Scenario B: Document Server Acquires Some Past Queries ................. 16
   6.3 Scenario E: Document Server Colludes with the Access Manager .......... 17
7. CASE STUDY ........................................................................ 18
   7.1 Description of Prototype .................................................. 18
   7.2 Apply Clustering on the Document Vectors .................................. 18
   7.3 Partition the Large Clusters. ............................................... 19
   7.3 Index the Partitions. ....................................................... 20
   7.4 Retrieval through the Index. ............................................... 20
8. CONCLUSION ........................................................................ 23
9. REFERENCES ......................................................................... 24
List of Figures

Figure 1 Term Document Matrix ................................................................. 5
Figure 2 DRM ......................................................................................... 9
Figure 3 Solution Approach ................................................................. 11
Figure 4 ............................................................................................... 12
Figure 5 Query processing ................................................................ 14
Figure 6 Clustering .............................................................................. 18
Figure 7 ............................................................................................... 19
Figure 8 ............................................................................................... 20
Figure 9 ............................................................................................... 21
Figure 10 Precision Recall ................................................................. 22
1. INTRODUCTION

Today’s text retrieval systems must have access to the plaintext corpus and queries. To illustrate, Figure 1 shows a set of example documents, with the corresponding term-document matrix1 representation that is typically used to facilitate retrieval. In the matrix, the rows correspond to the keywords, while the columns represent the documents. At runtime, each search query is transformed into the same representation as the documents, in order to match against the columns in the matrix. The nonzero cells indicate clearly what terms appear in each document or query. In fact, from the matrix the system is able to reconstruct the exact term frequencies in every user query and retrieved document. With the queries and retrieved documents in plain view of the text retrieval system, users must trust it to not abuse the privilege. This arrangement is not always desirable. Users are increasingly wary that their queries could disclose personal or confidential information to the search server.

<table>
<thead>
<tr>
<th>Term</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
<th>$d_6$</th>
<th>$d_7$</th>
<th>$d_8$</th>
<th>$d_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>.58</td>
<td>0</td>
<td>0</td>
<td>.49</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>interface</td>
<td>.58</td>
<td>0</td>
<td>.57</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>computer</td>
<td>.58</td>
<td>.44</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>user</td>
<td>0</td>
<td>.32</td>
<td>.42</td>
<td>0</td>
<td>.46</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>system</td>
<td>0</td>
<td>.32</td>
<td>.42</td>
<td>.72</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>response</td>
<td>0</td>
<td>.44</td>
<td>0</td>
<td>0</td>
<td>.63</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>time</td>
<td>0</td>
<td>.44</td>
<td>0</td>
<td>0</td>
<td>.63</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EPS</td>
<td>0</td>
<td>0</td>
<td>.57</td>
<td>.49</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>survey</td>
<td>0</td>
<td>.44</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.63</td>
<td>0</td>
</tr>
<tr>
<td>trees</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>.71</td>
<td>.51</td>
<td>0</td>
</tr>
<tr>
<td>graph</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.71</td>
<td>.51</td>
<td>.46</td>
<td>0</td>
</tr>
<tr>
<td>minors</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.7</td>
<td>.63</td>
<td>0</td>
</tr>
</tbody>
</table>

*Figure 1 Term Document Matrix*
This scheme is able to limit any potential privacy leak by ensuring:

- **Low fidelity of reverse-engineered content.** The original term composition of a document/query, and hence its semantic content, cannot be accurately deduced from its suppressed representation.

- **High anonymity in search results.** From the suppressed representation of a document/query, it is not possible to accurately identify other similar documents

This prevents the document server from classifying the corpus around compromised documents/queries whose plaintext and suppressed representations are leaked. At the same time, our scheme preserves the usability of the text retrieval system. The original similarity ranking is maintained. At the end of the retrieval process, the user obtains the same result for her query as intended by the original retrieval mechanism.
2. BACKGROUND ON TEXT RETRIEVAL

One of the fundamental problems in text retrieval is to determine the relevance of a document to a given query. There are three classical text retrieval models—Boolean, vector space, and probabilistic. Here focus on vector space model

2.1 Keyword Matching

The Boolean model for text retrieval by keyword matching is based on set theory and Boolean algebra. Suppose $q$ is a query, in the form of a Boolean expression of keywords. Let $q_{dnf}$ be the disjunctive normal form of $q$, and $q_{cc}$ be a conjunct in $q_{dnf}$. The similarity between document $d$ and query $q$ is defined as:

$$S_{q,d} = \begin{cases} 1 & \text{if } \exists q_{cc} \in q_{dnf} \text{ such that } (\forall \text{ term } t \in q_{cc}, t \in d) \\ 0 & \text{otherwise.} \end{cases}$$

2.2 Similarity-Based Retrieval

Most text search engines rate the similarity of each document to a query based on these heuristics:

- Terms that appear in many documents are given less weight;
- Terms that appear many times in a document are given more weight;
- Documents that contain many terms are given less weight.

The heuristics are encapsulated in a similarity function, which uses some composition of the following statistical values:

In the vector space model, the score of a document $d$ with respect to a query $q$, similarity is defined to be the cosine of the angle between the corresponding document vector and query vector in multidimensional term space. A similarity function
Privacy preserving similarity based text retrieval

\[ S_{q,d} = \frac{\sum_t w_{d,t} \cdot w_{q,t}}{W_d \cdot W_q}, \]

- \( w_{q,t} = f_{q,t} \times \log \frac{N}{f_t} \)
- \( W_q = \sqrt{\sum_t w_{q,t}^2} \)
- \( w_{d,t} = f_{d,t} \times \log \frac{N}{f_t} \)
- \( W_d = \sqrt{\sum_t w_{d,t}^2} \)

- \( f_{d,t} \), the number of times that term \( t \) appears in document \( d \);
- \( f_{q,t} \), the number of times that term \( t \) appears in query \( q \);
- \( f_t \), the number of documents that contain term \( t \);
- \( N \), the number of documents in the corpus.
3. DOCUMENT RETRIEVAL MODEL

Step 1: The owner of the corpus creates the inverted index that is needed for query processing. Both the corpus and inverted index are suppressed partially through cryptographic techniques before distribution to the document servers.

Step 2: The keys for unlocking the index entries and document contents are kept by the access manager the owner does not need to remain online for query processing.

Step 3: The document server accepts suppressed user queries, and locates the most relevant documents through the index. The suppressed index forces false positives into the search result, so as to mask the true user intention. This scheme guarantees that there is no false negative; in other words, no legitimate result document will be missed out.
Step 4: The user recovers the suppressed index data relating to the returned documents, which enables her to prune away the false positives. The remaining documents are now ranked correctly according to their similarity scores.

Steps 5 and 6: The user then downloads the encrypted content of selected result documents (There could be several document servers, each hosting a different partition and/or replica of the index and the corpus.

The user interacts with the access manager to unlock the content of the selected documents a security mechanism is needed to ensure that the access manager cannot inspect the document contents after unlocking them. The access manager tracks the number of documents that it unlocks for each user, for accounting purposes.
4. SOLUTION APPROACH

This text retrieval scheme is designed for the vector space model. In this model, each document is represented as a vector in multidimensional term-space as illustrated in Figure 1. Let \( X = [x_{ij}] \) denote the mterm by n document matrix (\( m > n \)). The singular value decomposition (SVD) of \( X \) with rank \( r_0 \) is defined as: \( X = U \cdot \Sigma \cdot V^T \) such that \( U = [u_{il}] \) is the left singular matrix whose columns contain orthogonal, unit-length vectors; \( \Sigma \) is a diagonal matrix of Eigen values \( \text{diag}(\sigma_1, \ldots, \sigma_n) \) where \( \sigma_1 > \ldots > \sigma_{r_0} > \sigma_{r_0+1} = \ldots = \sigma_n = 0 \); and \( V = [v_{jl}] \) is the right singular matrix whose columns contain orthogonal, unit-length vectors. Above Figure illustrates the SVD procedure. With Latent Semantic Indexing, \( X \) is approximated by retaining only the \( r_1 \) most significant Eigen values in, along with the corresponding columns in \( U \) and \( V \). Typically, \( r_1 \ll r_0 \). If LSI is not applied, then \( r_1 = r_0 \). The documents are represented by the columns in \( V^T \). Continuing the running example in Figure 1, the corresponding with \( r_1 = 8 \) is given in Figure 4. Document retrieval entails finding the documents that are nearest neighbors of the query in the \( r_1 \)-factor space.
To provide data privacy, only $r_2$ ($\leq r_1$) of the factors (rows) in $VT$ are stored in plaintext on the document server. The values in the remaining $r_1 - r_2$ rows are encrypted, and can only be deciphered by authorized users. In general, the encrypted rows could be the leading or trailing ones, or they could be spread out. Moreover, all $r_1$ columns of $U$ and the $r_1$ Eigen values in $\Sigma$ are encrypted. The encrypted $U$ and $\Sigma$, along with the partially encrypted $V$, constitute the suppressed index. In Figures 3 and 4, the $r_2$ plaintext factors in $V^T$ are in a light shade, while the encrypted factors are in a darker shade.
5. DETAILED INDEXING AND RETRIEVAL

There are mainly three steps in detailed indexing and retrieval:

- Corpus preparation
- Query processing
- Retrieval of result document

5.1 CORPUS PREPARATION

1. Assign a randomly generated, but unique \( id_j \) to each document \( d_j \). Lock \( d_j \) into the form \((id_j, E_C(d_j, k_j), E_T(k_j, k_c))\).

2. Apply Singular Value Decomposition (SVD) to \( X \).

3. \( U \) and \( \Sigma \) are encrypted with a random key \( K_{SVD} \).

4. For each document or column vector \( d_j \) in \( V^T \), create a triplet:

\[
(d_j(r_2), E_C(id_j | \delta d_j, l_j), E_T(l_j, k_r))
\]

After corpus preparation, the locked documents, the encrypted \( U \) and \( \Sigma \), and the partially suppressed \( V \) are deposited on the document server.

\( k_{SVD} \), the corpus key \( k_c \) and index key \( k_r \) are kept by the access manager.

5.2 Query Processing

Given a user query, the document server should produce an answer that contains the \( k \) most similar documents. Since the denominator normalizes the query and document vectors to unit length, every document/query can be treated as a point on the surface of the unit hypersphere in \( r_1 \)-factor space, and the cosine similarity between two points is inversely proportional to the Euclidean distance between them. Therefore the similarity-
based retrieval is equivalent to finding the $k$ documents with the shortest Euclidean distance to the query in $r_1$-factor space. Query processing mechanism at the document server assembles a superset of the search result in two steps.

1. Initial search. Find the $k$NN($r_2$) in the $r_2$-factor space defined by the plaintext rows of $V_T$. Compute the maximum distance $\text{dist}$ between the $k$NN($r_2$) documents and the query in the full $r_1$-factor space.

2. Expanded search. Retrieve all the documents that reside up to a distance of $\text{dist}$ from the query in the reduced $r_2$-factor space. These documents are guaranteed to include $k$NN($r_1$), the actual result documents. $k$NN($r_1$) is likely to differ from $k$NN($r_2$).

![Figure 5 Query processing](image)

Figure 5 illustrates the query processing procedure. Suppose the corpus contains document vectors $d_1$ to $d_4$ (and many others that are farther from the query than those four). Moreover, $r_1 = 2$, $r_2 = 1$, and we need the two closest documents to the query $q$. With only the $x$-coordinates, the initial search yields $2\text{NN}_x = \{d_3, d_4\}$. $\text{dist}$ is then computed from the Euclidean distance between $d_4$ and $q$. The expanded result
5.3 Retrieval of Result Documents

Upon receiving the expanded result, the user interacts with the access manager to decipher the encrypted coordinates. With the complete document vectors, the user can now rank the documents to derive $k\text{NN}(r_1)$. To safeguard the documents, a straightforward approach is to protect their contents using the twin-lock mechanism that we alluded to earlier. The twin lock is realized from a commutative encryption function $ET(m, k)$ that encrypts a message $m$ with a key $k$ such that $ET(ET(m, k_1), k_2) = ET(ET(m, k_2), k_1)$. The encrypted content of a document is paired with a randomly generated, but unique document id to facilitate retrieval through an index.

Thus, each document $d_j$ is stored in the form $(id_j, ET(d_j, kc))$ where $kc$ is the encryption key for the corpus. The user downloads the result documents $\{id_1, ET(d_1, kc), \ldots\}$ in step 4. The encrypted content of each document $d_j$ in the result is encrypted again with the user’s key $k_u$: $ET(ET(d_j, kc), k_u) = ET(ET(d_j, k_u), k_u)$ – and sent to the access manager in step 5. The access manager decrypts the documents with $kc$, and the resulting $ET(d_j, k_u)$ are returned in step 6. The user then decrypts with $ku$ to recover $d_j$ while this straightforward approach works, exchanging the encrypted document contents in steps 5 and 6 would incur high transmission costs, especially if the documents are large. This can be mitigated by introducing a layer of indirection: The owner encrypts each document $d_j$ with a randomly generated document key $kj$, then locks $kj$ with $kc$. Thus, $d_j$ is stored as $(id_j, EC(d_j, kj), ET(kj, kc))$ where $EC$ is a conventional encryption function. The user can now send the encrypted document keys $ET(ET(kj, kc), k_u)$ in step 5, and in step 6 the access manager returns $ET(kj, k_u)$. The user then decrypts with $K_u$ to extract $kj$, and decrypts $EC(d_j, kj)$ to recover $d_j$. 

(demarcated by the dark outer circle with radius $dist$) now contains the actual result $2\text{NN}_{xy} = \{d_1, d_2\}$ (demarcated by the light inner circle). After decrypting the $y$-coordinates, the user can rank $d_1$ to $d_4$ and locate the top two matches correctly.
6 PROTOCOL ANALYSIS OF PRIVACY SAFEGUARDS

Consider the scenario where the document server and access manager may collude

6.1 Scenario A: Independent Document Server and Access Manager

Excluding the initial request for \( k_{SVD} \) to decrypt \( U \) and \( \Sigma \), the protocol involves two exchanges each with the document server and the access manager. The exchanges with the latter are protected by the user key \( k_u \) and hence are safe. The first exchange with the former involves suppressed representations of the query \( q(r_2) \) and candidate result documents \( d_j(r_2) \). Being vectors within a synthetic factor space that is derived through singular value decomposition, \( q(r_2) \) and \( d_j(r_2) \) convey no useful meaning on their own. The second exchange with the document server serves only to download encrypted documents. Therefore, the document server and the access manager, acting independently, are not able to compromise the privacy protection.

6.2 Scenario B: Document Server Acquires Some Past Queries

Suppose that the document server somehow gets hold of the plaintext and suppressed representation of some document, and poses it as a search query to the system. How precisely is the server able to identify other documents that have similar term compositions? This is quantified by the anonymity metric. Here denote the plaintext and suppressed representation of the query as \( q \) and \( q(r_2) \) respectively.

COROLLARY 1. The anonymity of a search result is the ratio of the number of documents in the expanded result over the number of documents requested. Intuitively, an anonymity level of \( x \) indicates that the genuine top- \( k \) matching documents are mixed with \( (x-1)k \) spurious result entries, on the average. The document server is not able to
discern the genuine documents from the spurious entries in the search result. Only the user, after deciphering the suppressed factors in the document vectors, can differentiate the two accurately.

6.3 Scenario E: Document Server Colludes with the Access Manager

Finally, we consider the scenario where the access manager may collude with the document server. In addition to the left singular matrix U and Eigen values Σ, now the document server can also inspect the content of the documents that each user downloads. A user may mitigate this threat by mixing her genuine document downloads among spurious documents. Since the document keys that are sent to the access manager for unlocking are secured with the user key, the user is still able to unlock only the genuine result documents and thus avoid paying for the spurious documents. Therefore this technique achieves anonymity, at the expense of download overheads. Furthermore, the owner may set up several document servers, so that the user can gather her result documents across multiple locations. This increases the difficulty of collusion attacks, as the adversary would have to seize control of multiple document servers concurrently in order to track the documents downloaded by each user.
7. CASE STUDY

7.1 Description of Prototype
The implementation of this scheme first uses the Lucene search engine to build an index from the corpus. Next, dump out Lucene’s index into a vocabulary of terms, along with an inverted list for each term. The inverted list enumerates the identity of each document that contains that term, together with the frequency of the term in that document. After pruning away those terms that appear in only one document, the term-document matrix $X$ is constructed. The document vectors are normalized, before using the SVD routine in MATLAB to generate the $U$, $\Sigma$, and $V$ matrices. Since relevance ranking in the vector space model is determined by the distance of the document vectors from the query vector, query processing with the suppressed document and query representations involves finding the $k$NNs ($k$ nearest neighbors), then all documents within a computed distance $dist$ of the query in the $r_2$-factor space. To carry out these retrieval operations efficiently, then construct an R-tree index over the document vectors in $V$.

![Figure 6 Clustering]

7.2 Apply Clustering on the Document Vectors.
We apply a clustering algorithm on the document vectors in $V$, based on their positions in the $r_2$- dimensional term space. The intent is to minimize the extent of the tree nodes,
by preventing outlier documents from stretching the nodes of the R-tree across empty regions of the factor space

7.3 Partition the Large Clusters.

The previous step is likely to produce a wide range of cluster sizes. For example, C1 contains only one document, whereas C4 holds 8 documents. We now split the clusters into partitions that contain at most \( f \) documents each, where \( f \) is the fan-out factor of the R-tree.

(a) If a cluster contains \( f \) or fewer documents, it is treated as one partition.

(b) If a cluster contains more than \( f \) documents, it is space-partitioned recursively till all the partitions have at most \( f \) documents each. The partitioning strategy follows the KDB-tree. Specifically, we section a large partition along the first of the \( r2 \) plaintext factors into smaller partitions containing equal number of documents, then the new partitions along the second factor and so on, till the partitions are small enough.

Figure 9 shows how the clusters are partitioned for \( f = 2 \). (In practice, \( f \) is typically set to 100 or higher.) Clusters C2, C3 and C4 are first sectioned along the x-axis, followed by the y-axis, into 12 partitions that contain at most two documents each.

![Figure 7](image-url)
7.3 Index the Partitions.

Next, a Minimum Bounding Region (MBR) is defined for each partition; this MBR is the smallest hyper-rectangle that encloses all the documents within the partition. The partitions are then bulkloaded into the R-tree. The dimensionality of the R-tree is the maximum number of factors $r_3$ ($r_3 \leq r_2$) that step 2 takes to carve all the clusters into partitions that contain at most $f$ documents.

![Figure 8](image_url)

7.4 Retrieval through the Index.

Suppose that the R-tree $R$ constructed above indexes the first $r_3$ of the $r_2$ plaintext factors in $V$ & need to apply the procedure twice to generate the query result

- Phase 1: Locate the $k$ documents that are nearest the query in the $r_3$-dimensional space, $k\text{NN}(r_3)$. Compute the maximum distance $dist1$ between these $k\text{NN}(r_3)$ documents and the query in the $r_2$-dimensional space.
- Phase 2: Locate all the documents that are up to a distance of $dist1$ from the query in the $r_3$-dimensional space, using the R-tree. Rank these documents by their similarity to the query in the $r_2$-dimensional space, then identify $k\text{NN}(r_2)$.
Compute the maximum distance dist2 between these kNN(r2) documents and the query in the r1-dimensional space.

- Phase 3: Retrieve all the documents that reside up to a distance of dist2 from the query in the r3-dimensional space, using the R-tree. From these documents, prune away those that are beyond dist2 from the query in the r2-dimensional space. The remaining documents will include kNN(r1), the actual result documents.

Here use two corpora for our experiments. The first (WSJ) contains articles published in the Wall Street Journal from July to September of 1990. The second corpus (RCV1) is the training set of the Reuters Corpus Volume 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>WSJ</th>
<th>RCV1</th>
</tr>
</thead>
<tbody>
<tr>
<td>n: # of documents in the dataset</td>
<td>10,123</td>
<td>23,149</td>
</tr>
<tr>
<td>m: # of terms in the dataset</td>
<td>46,493</td>
<td>47,236</td>
</tr>
<tr>
<td>r1: Default # of coordinates kept by LSI</td>
<td>1,500</td>
<td>1,500</td>
</tr>
<tr>
<td>mask%: Ratio of r1 coordinates that are encrypted (= 1 – r2/r1)</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

The evaluation metrics for our experiments include the two basic measures of retrieval effectiveness, precision and recall.

\[
\text{precision} = \frac{\# \text{ of relevant documents in the search result}}{\text{total } \# \text{ of documents in the search result}}, \quad \text{recall} = \frac{\# \text{ of relevant documents in the search result}}{\text{total } \# \text{ of relevant documents}}.
\]

The privacy metrics are the following.

- The fidelity of the approximated corpus and user queries that could be reverse-engineered from the suppressed V(r2) and q(r2). Fidelity can be calculated directly from the Eigen values of the term-document matrix X.

- The anonymity accorded to the search results, which is derived by in turn treating each document as a query and measuring the anonymity of the query result, then averaging across all the documents in the corpus.
Figure 10 Precision Recall
8. CONCLUSION

While the usage of text retrieval systems has undergone tremendous growth in the past decade, development of security mechanisms to safeguard the privacy of users of such systems has not kept pace. This article introduces a solution that enables a document server to perform similarity-based text retrieval while protecting user privacy. Based on the vector space model, the query and document representation that the document server relies upon for query processing discloses no information about the search queries. Even in the event that the document server manages to acquire extra information through collusion with other parties, our solution is still able to limit any privacy leaks. Moreover, the privacy protection is achieved without altering the relevance ranking of the original retrieval algorithm.
9. REFERENCES


