2-2010

Privacy-Preserving Similarity-Based Text Retrieval

Hwee Hwa PANG
Singapore Management University, hhpang@smu.edu.sg

Jialie SHEN
Singapore Management University, jlshen@smu.edu.sg

Ramayya Krishnan
Carnegie Mellon University

DOI: https://doi.org/10.1145/1667067.1667071

Citation

PANG, Hwee Hwa; SHEN, Jialie; and Krishnan, Ramayya. Privacy-Preserving Similarity-Based Text Retrieval. (2010). *ACM Transactions on Internet Technology*. 10, (1),. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/220

Privacy-Preserving Similarity-Based Text Retrieval

HWEEHWA PANG and JIALIE SHEN
Singapore Management University
and
RAMAYYA KRISHNAN
Carnegie Mellon University

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. In this article, we introduce a privacy-preserving, similarity-based text retrieval scheme that (a) prevents the server from accurately reconstructing the term composition of queries and documents, and (b) anonymizes the search results from unauthorized observers. At the same time, our 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.

Categories and Subject Descriptors: H.3 [**Information Systems**]: Information Storage and Retrieval

General Terms: Security

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

H. H. Pang is supported by a research grant (C220/MSS6C002) from the Singapore Management University.

Authors' addresses: H. H. Pang, School of Information Systems, Singapore Management University; email: hhpang@smu.edu.sg; J. Shen, School of Information Systems, Singapore Management University; R. Krishnan, School of Information Systems and Management, Heinz College, Carnegie Mellon University, Pittsburgh, PA.

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 matrix[1] 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. Public search engines have been reported to track user queries, in order to push targeted advertisements [Hansell 2006]. In a potent demonstration of the privacy risk posed by search histories, AOL recently released its Web log data, only to withdraw it soon after when it was shown that detailed user profiles could be constructed from the data [Barbaro and Zeller 2006]. User privacy concerns have also been cited as a factor that hinders the adoption of personalized search [Shen et al. 2007].

Consider the deployment scenario depicted in Figure 2. Suppose a content provider like MicroPatent (the "owner") maintains a patent and trademark database that is deployed onto several online document servers. The paid subscribers (the "user") include patent researchers who search for information to assess the value of potential companies to invest in, or to size up the competitive landscape for new product development. These users would want to ensure that their search activities are confidential, especially to their competitors.

Naturally, the administrator of an online document service would employ a combination of security safeguards, such as firewalls and intrusion detection. Notwithstanding that, document servers that are situated in a seemingly well-guarded network often can still be infiltrated through a multistep intrusion, in which each step paves the way for the next attack [Wang et al. 2006]. Indeed, the number of successful attacks on online services has multiplied over the past decade. The perpetrators include external hackers and, worse still, insiders. The victims range from government and large companies to e-business sites that we would expect to have been professionally administered and secured. This evidence suggests that it is very difficult to guarantee the security of all the document servers over extended periods of time. In the event that a document server is compromised, the intruder would be able to monitor users' search queries as well as the content of the documents retrieved.

[1]The formulation of the term-document matrix is explained in Section 2.2. As the matrix is sparse, it is usually stored as inverted term lists to conserve space [Zobel and Moffat 2006].

Document	Content
$\mathbf{d}_1$	Human machine interface for Lab ABC computer applications
$\mathbf{d}_2$	A survey of user opinion of computer system response time
$\mathbf{d}_3$	The EPS user interface management system
$\mathbf{d}_4$	System and human system engineering testing of EPS
$\mathbf{d}_5$	Relation of user-perceived response time to error measurement
$\mathbf{d}_6$	The generation of random, binary, unordered trees
$\mathbf{d}_7$	The intersection graph of paths in trees
$\mathbf{d}_8$	Graph minors IV: Widths of trees and well-quasi-ordering
$\mathbf{d}_9$	Graph minors: A survey

Term	Documents								
	$\mathbf{d}_1$	$\mathbf{d}_2$	$\mathbf{d}_3$	$\mathbf{d}_4$	$\mathbf{d}_5$	$\mathbf{d}_6$	$\mathbf{d}_7$	$\mathbf{d}_8$	$\mathbf{d}_9$
human	.58	0	0	.49	0	0	0	0	0
interface	.58	0	.57	0	0	0	0	0	0
computer	.58	.44	0	0	0	0	0	0	0
user	0	.32	.42	0	.46	0	0	0	0
system	0	.32	.42	.72	0	0	0	0	0
response	0	.44	0	0	.63	0	0	0	0
time	0	.44	0	0	.63	0	0	0	0
EPS	0	0	.57	.49	0	0	0	0	0
survey	0	.44	0	0	0	0	0	0	.63
trees	0	0	0	0	0	1	.71	.51	0
graph	0	0	0	0	0	0	.71	.51	.46
minors	0	0	0	0	0	0	0	.7	.63

Fig. 1. Sample corpus (adapted from Deerwester et al. [1990]).

There are a number of studies on privacy-preserving similarity-based text retrieval in the literature. TrackMeNot[2] attempts to bury the genuine user queries among random "ghost" queries. Shen et al. [2007] examines user privacy issues in personalized search engines; however, the solutions offered in the paper do not include any for securing the document server itself. Jiang et al. [2008] describe a scheme for a user to compute similarity scores for all the documents on the server without it knowing what the query is; the computation costs for the user and the server are proportional to the number of documents, hence the scheme is not intended for search engines that need to support large corpora. In addition, there are exact keyword-matching schemes for encrypted data (e.g., Agrawal et al. [2004a], Damiani et al. [2003], Hacigumus et al. [2002], Song et al. [2000]), which do not extend to similarity-based matching.

Contributions. In this article, we present the first scheme for a document server to perform similarity-based text retrieval while protecting the privacy of users' search activities; in particular, we aim to safeguard the content (or

[2]http://mrl.nyu.edu/dhower/trackmenot.

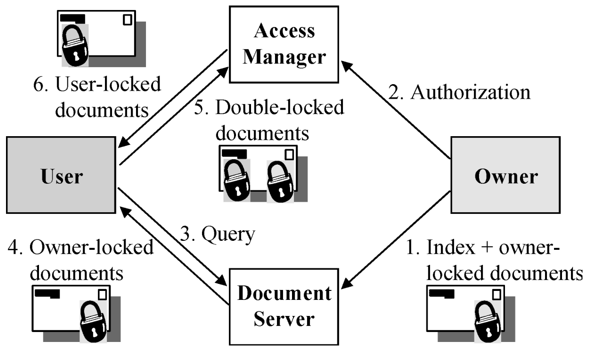


Fig. 2. Document retrieval model.

semantics) of the user queries and the retrieved documents. We target the *vector space model* [Salton 1989] that returns, for each user query, the documents that are most similar to the query terms. In this model, each document is represented as a vector in term-space (i.e., a column in the term-document matrix in Figure 1), and queries are treated like documents. Moreover, relevance ranking is determined by the distance of the document vectors from the query vector. Thus, document retrieval entails finding documents that have the shortest distances to the query vector.

Based on the system model in Figure 2, our scheme enables the document server to process search queries with just a suppressed representation of the queries and documents, *without interfering with the relevance ranking of the retrieval algorithm.* The suppressed representation hides the semantics of the queries from the document server, thus protecting user privacy. Even in the event that the document server manages to acquire extra information through collusion with other parties, our 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.

—*Usage accounting.* The number of documents that each user opens can be accounted for. This is flexible enough to accommodate different usage models, for example pay-per-use and fixed-price subscription.

The rest of this article is organized as follows. The next section gives background on two conventional text retrieval models, including the vector space model that enables similarity-based retrieval, while Section 3 reviews related work. Section 4 presents our text retrieval scheme. The privacy safeguards provided by the scheme are analyzed in Section 5. Following that, Section 6 describes our prototype implementation, and provides empirical evidence of the effectiveness of our privacy safeguards. Finally, Section 7 concludes the article.

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. Over the last forty years, there have been numerous studies on this problem. There are three classical text retrieval models—Boolean, vector space, and probabilistic. The review in this section covers the first two. The Boolean model underlies keyword-matching systems, to which existing privacy-preserving methods are applicable. The vector space model enables similarity-based retrieval, and is the target of our solution in this article. As far as we know, there is no privacy-preserving scheme in the literature for the probabilistic model, so we will skip that 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:

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

This similarity function produces a binary decision without any notion of ranking. In other words, a document either matches the query or it does not; there is no degree of relevance that differentiates two matching documents. The limitation hinders good retrieval performance when users are not familiar with the exact terminology in the documents, and is especially problematic with casual users or large/heterogeneous corpora. This drawback of the Boolean model prompted the development of similarity-based retrieval systems.

2.2 Similarity-Based Retrieval

As explained in Zobel and Moffat [2006], 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; and

—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:

—$f_{d,t}$, 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.

In the vector space model, the score of a document d with respect to a query q, $S_{q,d}$, is defined to be the cosine of the angle between the corresponding document vector and query vector in multidimensional term space. A similarity function that is found to work well in practice is:

$$S_{q,d} = \frac{\sum_t w_{d,t} \cdot w_{q,t}}{W_d \cdot W_q}, \tag{1}$$

where

—$w_{q,t} = f_{q,t} \times \log \frac{N}{f_t}$
—$W_q = \sqrt{\sum_t w_{q,t}^2}$
—$w_{d,t} = f_{d,t} \times \log \frac{N}{f_t}$
—$W_d = \sqrt{\sum_t w_{d,t}^2}$.

The example in Figure 1 follows this formulation, with each cell in the term-document matrix set to $w_{d,t}/W_d$.

Most modern search engines are based on the vector space model rather than Boolean keyword matching, as the former allows result documents to be ranked by their similarity to the query. We thus adopt the vector space model in this article.

3. RELATED WORK

In this section, we focus on related work on privacy in text retrieval, data privacy, anonymous communication, and access control, which are most relevant to our study.

3.1 Privacy in Text Retrieval

Klein et al first examined the problem of protecting the corpus in a text retrieval system in Klein et al. [1989]. Their scheme compresses the documents through Huffman encoding, so that they appear like random bit-strings to unauthorized observers. However, the index—dictionary and concordance—that is used for text searches is compressed but not encrypted. The challenge of compressing the documents and index has received further attention, for example in Bookstein et al. [1992], Long [2002], Zobel and Moffat [1995]. These compression schemes do not provide the basis for the solution that we seek, because without encryption the privacy safeguards cannot be enforced when the document server itself becomes compromised by attackers.

TrackMeNot, a browser extension, protects user queries from public search engines by hiding a user's genuine search queries among randomly generated "ghost" queries. This is similar to k-anonymity protection (to be described shortly), and is useful for masking individual user queries. If a user submits multiple queries during a session, however, the correlation among the genuine queries could differentiate them from the random ghost queries. This is especially so for paid document services like the patent search application described in Section 1, where queries that do not lead to paid document downloads are probably not interesting to the user and so can safely be filtered out. Yet, it would be too expensive for the user to pay for documents retrieved by the ghost queries. Finally, if employed widely, the overhead imposed by all the ghost queries on the network infrastructure and search engines could significantly degrade query response times.

Shen et al. [2007] suggested that user privacy may be protected by pushing the search index and query processing to a trusted third party, or by legally compelling the search engine to "forget" the user activities right after they are served. However, the risk of privacy disclosure when the trusted third party or search engine is infiltrated remains an open problem.

Jiang et al. [2008], introduced a scheme for the user to compute a similarity score between a given query and each document on a server. To safeguard the query, it is presented to the server as a vector of encrypted term weights. The server then combines the query vector with the document vectors in turn to produce a list of encrypted scores for the user. As this procedure has to be carried out on every document on the server, it could be too expensive for search engines that need to support large corpora. To achieve better scalability, search engines must be able to find the most relevant documents to a query by examining just a small subset of the corpora.

3.2 Data Privacy through Encryption

Cryptographic techniques for an untrusted server to perform keyword-matching over encrypted data, without any knowledge of its plaintext, have been proposed in Song et al. [2000], Kurosawa and Ogata [2004], Freedman et al. [2005], Bethencourt et al. [2006], and Ostrovsky and Skeith [2007]. While they can be used to implement keyword-matching retrieval, these techniques do not apply to the vector space model of text retrieval because the similarity score of a document cannot be computed from its ciphertext.

Privacy-preserving retrieval has also been studied in the context of databases. For example, Agrawal et al. [2004a], Damiani et al. [2003], and Hacigumus et al. [2002] described how the tables and indices in a relational database can be encrypted to still allow the execution of SQL queries. The techniques can be adapted to perform keyword matching for text documents, but again are not applicable to a text retrieval system that needs to support similarity-based retrieval.

Domingo-Ferrer et al. [2008] proposed a protocol for users to access collaborating users' resources through private relationships in a social network. The relationships and trust levels of the collaborating users are safeguarded through homomorphic encryption techniques. In our work here, we employ similar encryption techniques to safeguard the identity of the documents that users retrieve.

Besides data encryption, specialized secure storage schemes have been proposed that provide additional privacy guarantees: Steganographic file systems (e.g., Anderson et al. [1998], Pang et al. [2004]) allow a data owner to plausibly deny the existence of protected data, while private information retrieval (e.g., Chor et al. [1995], Wang et al. [2007]) prevents the server from finding out which object is being retrieved. These two types of storage schemes rely on encryption to protect the data, so they too do not allow the server to compute the similarity scores between the documents and the query.

3.3 Data Privacy through Hashing

Goh [2003] proposed a secure index scheme, called Z-IDX. A searcher with a trapdoor for some given word can test its existence in the index in $O(1)$ time. Trapdoors are generated with a secret key, and are built on Bloom filters [Bloom 1970] and pseudo random functions. Without the secret key and legitimate trapdoors, an adversary can deduce no information from the index. One of the stated applications for Z-IDX is in keyword matching on encrypted documents.

Bawa et al. [2003] proposed to construct a privacy-preserving index (PPI) on a set of documents from different providers. Each provider summarizes the terms in its shared content through a bit vector such as a Bloom filter [Bloom 1970]. At runtime, the index server accepts a query and returns a shortlist of providers that may contain matching documents. If the bit vector of a provider gives a negative for the query terms, the provider is guaranteed to hold no matching documents; otherwise, there is a possibility that matching documents can be found with the provider. The searcher then queries the shortlisted providers directly to request matching documents. The primary differences of PPI from our work are (a) PPI does not protect the privacy of the user queries, (b) the PPI scheme targets keyword matching, rather than similarity-based retrieval, and (c) PPI affects the precision-recall effectiveness of the original retrieval mechanism.

3.4 Data Privacy through Anonymity

In statistical databases, k-anonymity has been proposed as an alternative privacy measure to encryption. A database is k-anonymous if every record in it is indistinguishable from at least $k-1$ other records with respect to the accessible attributes [Samarati 2001; Sweeney 2002]. The problem of k-anonymization is NP-hard [Meyerson and Williams 2004]. To tackle the computation complexity, Agrawal et al. [2004b] provided approximation algorithms for generating k-anonymous tables, while Zhong et al. [2005] developed a protocol for generating k-anonymous tables in distributed environments. Machanavajjhala et al. [2006] demonstrated that k-anonymity still allows information leakage when there is lack of diversity in sensitive attributes; to counter that, the ℓ-Diversity was proposed as an alternative privacy measure. However, Xiao and Tao [2006] showed that an adversary can find out sensitive information with 100% confidence even in Machanavajjhala's scheme.

k-Anonymity has also been used in protecting user privacy in location-based services [Gruteser and Grunwald 2003; Kalnis et al. 2007]. A common technique there is to artificially enlarge a user's region of interest to envelop $k - 1$ other users, so that the server is not able to pinpoint the user who issues a given query. The technique could be adapted for Boolean text search, by padding a user query with spurious terms so that the server is not able to identify the exact terms that characterize the user's intention. However, in the vector space model, such a padded query would point to a different location in the term space. Without precise knowledge of the document distribution in the term space, the user has no basis for deciding how many more documents need to be retrieved without missing any legitimate result documents for her genuine query. Therefore, this technique cannot be applied to the vector space model.

3.5 Data Privacy through Noise Addition

Another technique for achieving data privacy in statistical databases is to introduce noise to the data collection. By perturbing the data in a controlled manner, it is possible to prevent accurate estimation of individual observations while preserving certain statistical properties (e.g. mean, covariance, variance) of the overall database (e.g., Kim [1986], Domingo-Ferrer et al. [2004]). However, it is not clear how this technique can be applied to a text corpus without affecting the similarity scoring function.

A number of related techniques have also been studied in the context of structured data in the information systems literature [Duncan et al. 2001; Gopal et al. 2002; Menon and Sarkar 2007; Kadane et al. 2006]. However, the applicability of these techniques to our text corpus and text retrieval setting remains an open question.

3.6 Anonymous Communication

Mechanisms for anonymous communication are also relevant to our work. Such mechanisms enable a user to submit queries to the document server without being identified, and is complementary to our proposal here for safeguarding the content of user queries and documents. Studies on anonymous communication abound, from mix-net [Chaum 1981], to Onion Routing [Goldschlag et al. 1999], to the recent Tor system [Dingledine et al. 2004].

3.7 Access Control

Access control concerns the enforcement of which user has what access rights over any data or resource. There is a rich body of work on access control, from model [Sandhu et al. 1996] to system design [Jajodia et al. 2001] to applications [Agrawal et al. 2002]. Access control is complementary to data privacy; the latter provides protection in situations where the adversary manages to circumvent the access control mechanism and gain control of the system internals or underlying storage.

4. TEXT RETRIEVAL SCHEME

The notion of user privacy for search engine was studied recently in Shen et al. [2007], which defined four levels of privacy protection. Roughly, level 1 replaces the user identity with a pseudo identity, but exposes the user information needs

Table I. Notations

Symbol	Definition
$\mathbf{X} = [x_{ij}]$	m term by n document matrix ($= \mathbf{U}\cdot\Sigma\cdot\mathbf{V}^T$)
$\mathbf{U} = [u_{il}]$	Left singular matrix
$\Sigma = diag(\sigma_i)$	Diagonal matrix of Eigenvalues
$\mathbf{V} = [v_{jl}]$	Right singular matrix
r_0	# of non-zero Eigenvalues in Σ
r_1	# of factors kept by LSI
r_2	# of factors left in plaintext
E_T	Commutative encryption function
E_C	Conventional encryption function, e.g. AES
k_c	Corpus encryption key
k_r	Index encryption key
k_u	User encryption key
k_j	Encryption key for document d_j
kNN	k Nearest Neighbors requested by the query
f	Fan-out of the R-tree

(as represented by the query text and the result documents that are retrieved). Level 2 masks the individual identities, so the server knows only the group identity behind a search. Level 3 removes all identity information, while level 4 hides even the information needs.

The focus of our scheme is to safeguard the user queries and retrieved documents, so as to mask the user information needs from any adversary. A combination of our scheme with an anonymous communication protocol to protect the user identities will attain level-4 privacy. Anonymous communication has been studied extensively elsewhere (e.g. Chaum [1981], Goldschlag et al. [1999], Dingledine et al. [2004]), and is beyond the scope of our work. In this section, we first define our system model and threat scenarios, then introduce our privacy-preserving retrieval scheme. The notations used are summarized in Table I.

4.1 Problem Definition

Our objective is to design a scheme for a text retrieval system, built on the vector space model, to perform similarity-based retrieval while safeguarding the privacy of user activities, particularly the user queries and retrieved documents.

Our scheme follows the interactions depicted in Figure 2.

—The owner of the corpus creates the inverted index [Zobel and Moffat 2006] 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 1). The keys for unlocking the index entries and document contents are kept by the access manager (step 2). The owner does not need to remain online for query processing.

—The document server accepts suppressed user queries (step 3), 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.

Our scheme guarantees that there is no false negative; in other words, no legitimate result document will be missed out.

—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. The user then downloads the encrypted content of selected result documents (step 4). 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 (steps 5 and 6). 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.

The primary threat is that an adversary may gain control of the document server, and collude with some users and even the access manager in order to monitor what other users are searching for. Against such an adversary, our scheme provides the following privacy protection:

—The adversary must not be able to deduce accurately the term composition, and hence the semantic content, of the documents and queries. We quantify this potential information leakage with the *fidelity* metric.

—The adversary must not be able to accurately identify other documents that are similar to any given document or query. This prevents the adversary from classifying the corpus around compromised documents/queries whose plaintext and suppressed representations have been leaked. Here the protection is achieved through *anonymity*.

Definition 1. As explained in Section 2.2, the term composition of any document or query is expressed as a vector. Let **X** be a matrix in which the columns correspond to the document and query vectors that we wish to protect. The Frobenius norm of **X**, defined as $||\mathbf{X}||_F = \sqrt{\sum_{i=1}^{r_1} diag(\mathbf{X}^T\mathbf{X})}$, is the Euclidean length of all the document vectors in **X**.

Furthermore, let $\hat{\mathbf{X}}$ denote the corresponding document and query vectors that the adversary is able to deduce. The difference between the document/query vectors in **X** and $\hat{\mathbf{X}}$ is measured by $||\mathbf{X} - \hat{\mathbf{X}}||_F$.

The *fidelity* of the estimated $\hat{\mathbf{X}}$ is $1 - \frac{||\mathbf{X}-\hat{\mathbf{X}}||_F}{||\mathbf{X}||_F}$.

A fidelity of 1 indicates that the adversary is able to reconstruct **X** perfectly, a situation that we wish to prevent. A fidelity of 0 means that the adversary has zero knowledge of **X**; while such an assurance might appear desirable from a purely security standpoint, it severely cripples the retrieval function of the document server at the same time. Intuitively, the less is known about the corpus and query, the less accurately can the document server filter out irrelevant documents from the search result. In practice, we need to balance between preserving retrieval efficiency and limiting privacy disclosure, taking into account

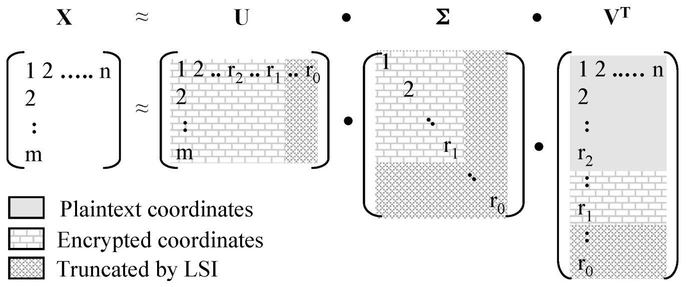


Fig. 3. Partially suppressed representation.

the likelihood of such disclosure (i.e., the probability of user-document server collusion).

Definition 2. In a search result with an *anonymity* level of x, the genuine top-k matching documents are mixed with $(x-1)k$ other entries; the adversary is not able to tell the two apart.

Anonymity requires the search result to be padded with irrelevant documents, and is proportional to the cost imposed by our scheme. Therefore our aim is to achieve low fidelity while staying within certain target anonymity ranges. In addition to privacy protection, our scheme must meet usability requirements:

(1) The final search results obtained by the user must be the same as those intended by the original retrieval mechanism. This means that the document entries, their similarity scores, as well as the precision/recall of the search results are preserved.
(2) The number of documents that each user opens can be tracked.

4.2 Overview of Solution Approach

Our 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 $\mathbf{X} = [x_{ij}]$ denote the m term by n document matrix $(m > n)$. The singular value decomposition (SVD) of $\mathbf{X}$ with rank r_0 is defined as: $\mathbf{X} = \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^{\mathbf{T}}$ such that $\mathbf{U} = [u_{il}]$ is the left singular matrix whose columns contain orthogonal, unit-length vectors; Σ is a diagonal matrix of Eigenvalues $diag(\sigma_1, \ldots, \sigma_n)$ where $\sigma_1 > \ldots > \sigma_{r_0} > \sigma_{r_0+1} = \ldots = \sigma_n = 0$; and $\mathbf{V} = [v_{jl}]$ is the right singular matrix whose columns contain orthogonal, unit-length vectors. Moreover, $x_{ij} = \sum_{l=1}^{r_0} u_{il} \cdot \sigma_l \cdot v_{jl}$. Figure 3 illustrates the SVD procedure.

With Latent Semantic Indexing (LSI) [Deerwester et al. 1990], $\mathbf{X}$ is approximated by retaining only the r_1 most significant Eigenvalues in Σ, along with the corresponding columns in $\mathbf{U}$ and $\mathbf{V}$. Typically, $r_1 \ll r_0$. If LSI is not applied, then $r_1 = r_0$.

The documents are represented by the columns in $\mathbf{V}^T$. Continuing the running example in Figure 1, the corresponding $\mathbf{V}^T$ 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.

Factor	Documents								
	$\mathbf{d}_1$	$\mathbf{d}_2$	$\mathbf{d}_3$	$\mathbf{d}_4$	$\mathbf{d}_5$	$\mathbf{d}_6$	$\mathbf{d}_7$	$\mathbf{d}_8$	$\mathbf{d}_9$
1	-.038	-.116	-.052	-.043	-.059	-.440	-.552	-.573	-.387
2	.358	.508	.496	.432	.387	-.103	-.107	-.088	.040
3	.316	-.427	.336	.446	-.571	.155	.093	-.002	-.217
4	.085	-.036	-.015	.075	-.327	-.558	-.224	.211	.687
5	.850	.110	-.316	-.393	-.073	.053	.011	-.038	-.040
6	-.053	.334	-.675	.590	-.184	.154	-.011	-.153	.026
7	-.153	.406	.278	-.293	-.465	.447	-.205	-.350	.261
8	-.050	.159	.058	-.072	-.156	-.405	.763	-.440	.005

Fig. 4. $\mathbf{V}^T$ with $r_1 = 8$ and $r_2 = 4$ for the Sample Corpus.

To provide data privacy, only r_2 ($\leq r_1$) of the factors (rows) in $\mathbf{V}^T$ 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 $\mathbf{U}$ and the r_1 Eigenvalues in Σ are encrypted. The encrypted $\mathbf{U}$ and Σ, along with the partially encrypted $\mathbf{V}$, constitute the suppressed index. In Figures 3 and 4, the r_2 plaintext factors in $\mathbf{V}^T$ are in a light shade, while the encrypted factors are in a darker shade. We observe that by itself the suppressed $\mathbf{V}^T$ in Figure 4 bears little semblance to, and thus reveals no useful information to, any adversary about the original term-document $\mathbf{X}$ in Figure 1.

To submit a query, the user first obtains the decryption key for $\mathbf{U}$ and Σ from the access manager. The query (column) vector $\mathbf{q}_{(r1)} = (q_1, \ldots, q_{(r1)})$ is derived as $\mathbf{q}_{(r1)} = \Sigma^{-1} \cdot \mathbf{U}^{\mathbf{T}} \cdot \mathbf{q}$. $\mathbf{q}_{(r2)}$, comprising the r_2 plaintext coordinates in $\mathbf{q}_{(r1)}$, are sent to the document server.

Next, the document server assembles an intermediate result that is a superset of the result documents, using the r_2 plaintext rows of $\mathbf{V}^T$ as described in Section 4.3. The user then deciphers the $r_1 - r_2$ encrypted coordinates of the documents in the intermediate result, and with the additional coordinates prunes down to the actual search result.

A lower r_2 setting is expected to limit the document server to a poorer approximation of $\mathbf{X}$, thus reducing the fidelity of any reverse-engineered documents in case the document server manages to obtain $\mathbf{U}$ and Σ by colluding with some users. (We will examine this threat in detail in Section 5). In the running example in Figure 4, the fidelity drops from 0.92 at $r_2 = 8$ to 0.15 at $r_2 = 1$. At the same time, the intermediate results would include more false positives, thus raising anonymity. The price is that the user would incur more processing overheads in pruning false positives.

After identifying the result documents, the user may proceed to download their contents. To meet the requirements of usage accounting and preserving the privacy of documents retrieved by each user, we employ the cryptographic equivalent of a twin-lock: The document contents are locked by the owner before distribution. After downloading the target documents from the document server, the user applies her own lock on those documents, then sends them to the access manager to remove the owner's lock. This way, after the original lock by the owner is removed, the documents are still protected by the user's lock. The access manager is thus able to track the number of documents that are unlocked for each user, without peeking into their contents.

In addition to privacy, the user is likely to be concerned about the *integrity* of the text retrieval process—that the query results generated by the document server are correct, and have not been tampered with. Integrity of search results has been addressed elsewhere (e.g., Pang and Mouratidis [2008]), and is beyond the scope of this paper.

4.3 Query Processing

Given a user query, the document server should produce an answer that contains the k most similar documents, according to the similarity function in Formula (1). 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.

Our query processing mechanism at the document server assembles a superset of the search result in two steps.

(1) *Initial search*. Find the $k\mathrm{NN}_{(r_2)}$ in the r_2-factor space defined by the plaintext rows of $\mathbf{V}^T$. Compute the maximum distance *dist* between the $k\mathrm{NN}_{(r_2)}$ documents and the query in the full r_1-factor space.
(2) *Expanded search*. Retrieve all the documents that reside up to a distance[3] of *dist* from the query in the reduced r_2-factor space. These documents are guaranteed to include $k\mathrm{NN}_{(r_1)}$, the actual result documents. $k\mathrm{NN}_{(r_1)}$ is likely to differ from $k\mathrm{NN}_{(r_2)}$.

This search procedure is conducted in the reduced r_1-factor space (produced through truncated SVD), not in the original term space. Truncated SVD is a common technique in text document indexing and retrieval. Previous studies (e.g., Kolda and O'Leary [1998], He et al. [2004], Yan et al. [2008]) have reported that the reduced factor space tends to bring together terms and documents based on co-occurrence. Therefore expanding the search distance in step 2 has the effect of pulling in terms and documents that have lower co-occurrence with the query terms. This, together with the polysemy effect (where a word has multiple meanings), introduce diversity into the search result and lower its fidelity to the adversary, as shown in Section 6.

We now prove that the search procedure is correct. Let $\mathbf{q}_{(r_1)} = (q_1, \ldots, q_{r_1})$ and $\mathbf{q}_{(r_2)} = (q_1, \ldots, q_{r_2})$ denote the full and suppressed query vectors. $\mathbf{q}_{(r_1)} = \mathbf{q}_{(r_2)} + \delta\mathbf{q}$

[3]We focus on the Euclidean distance here, though our solution extends easily to the L_p or Minkowski metrics.

where $\delta\mathbf{q}$ is in an orthogonal subspace from $\mathbf{q}_{(r_2)}$, such that $\sqrt{|\mathbf{q}_{(r_2)}|^2 + |\delta\mathbf{q}|^2} = |\mathbf{q}_{(r_1)}|$. Thus $|\delta\mathbf{q}|^2 = |\mathbf{q}_{(r1)}|^2 - \sum_{l=1}^{r_2} q_l^2$.

Similarly, let $\mathbf{d}_{j(r_1)} = (d_{1j}, \ldots, d_{r_1 j})$ and $\mathbf{d}_{j(r_2)} = (d_{1j}, \ldots, d_{r_2 j})$ denote the full and suppressed vectors for a document $\mathbf{d}_j$ in $k\mathrm{NN}_{(r_2)}$. $\mathbf{d}_{j(r_1)} = \mathbf{d}_{j(r_2)} + \delta\mathbf{d}_j$ where $\delta\mathbf{d}_j$ is orthogonal to $\mathbf{d}_{j(r_2)}$, and $|\delta\mathbf{d}_j|^2 = |\mathbf{d}_{j(r1)}|^2 - \sum_{l=1}^{r_2} d_{lj}^2$.

By definition, $\mathbf{d}_{j(r_1)} - \mathbf{q}_{(r_1)} = \mathbf{d}_{j(r_2)} + \delta\mathbf{d}_j - \mathbf{q}_{(r_2)} - \delta\mathbf{q} = (\mathbf{d}_{j(r_2)} - \mathbf{q}_{(r_2)}) + (\delta\mathbf{d}_j - \delta\mathbf{q})$. $\delta\mathbf{d}_j - \delta\mathbf{q}$ is longest when $\delta\mathbf{d}_j$ and $\delta\mathbf{q}$ are exactly in opposite directions. Thus,

$$\left|\mathbf{d}_{j(r_1)} - \mathbf{q}_{(r_1)}\right|^2 \le \left|\mathbf{d}_{j(r_2)} - \mathbf{q}_{(r_2)}\right|^2 + \left(\sqrt{|\mathbf{d}_{j(r_1)}|^2 - \sum_{l=1}^{r_2} d_{lj}^2} + \sqrt{|\mathbf{q}_{(r_1)}|^2 - \sum_{l=1}^{r_2} q_l^2}\right)^2.$$

The maximum distance between the $k\mathrm{NN}_{(r_1)}$ documents and the query in the full r_1-factor space is:

$$dist = \max_j \left[|\mathbf{d}_{j(r_2)} - \mathbf{q}_{(r_2)}|^2 + \left(\sqrt{|\mathbf{d}_{j(r_1)}|^2 - \sum_{l=1}^{r_2} d_{lj}^2} + \sqrt{|\mathbf{q}_{(r_1)}|^2 - \sum_{l=1}^{r_2} q_l^2}\right)^2\right]^{\frac{1}{2}} \tag{2}$$

for all $\mathbf{d}_j \in k\mathrm{NN}_{(r_2)}$. By including the value of $|\mathbf{q}_{(r1)}|$ in the query structure and storing the $|\mathbf{d}_{j(r1)}|$ values on the document server, it has enough information to compute *dist* with just the plaintext rows of $\mathbf{V}^T$. The expanded result that is returned to the user contains the document vectors, including their plaintext and encrypted coordinates.

LEMMA 1. *Any document* $\mathbf{d}_j$ *that is not retrieved in the expanded search cannot be in* $k\mathit{NN}_{(r_1)}$*, the actual result.*

PROOF. Given that $\mathbf{d}_j$ is excluded from the expanded search for documents that are up to *dist* from the query, $|\mathbf{d}_{j(r_2)} - \mathbf{q}_{(r_2)}| > dist$. It follows that $|\mathbf{d}_{j(r_1)} - \mathbf{q}_{(r_1)}| > dist$.

Since the expanded search includes the $k\mathrm{NN}_{(r_2)}$ from the initial search, there are already at least k documents that are within a distance of *dist* from the query in the full r_1-factor space. Therefore $\mathbf{d}_j \notin k\mathrm{NN}_{(r_1)}$. □

LEMMA 2. *The proposed retrieval scheme correctly produces the k most relevant documents to the query.*

PROOF. Lemma 1 proves that all the k most relevant documents are included in the expanded result. After decrypting the $r_1 - r_2$ encrypted coordinates, the user can order the documents in the expanded result with all the r_1 coordinates. By definition, the k most relevant documents must be at the top of the ordered list. □

Figure 5 illustrates the query processing procedure. Suppose the corpus contains document vectors $\mathbf{d}_1$ to $\mathbf{d}_4$ (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 $\mathbf{q}$. With only the x-coordinates, the initial search yields $2\mathrm{NN}_x = \{\mathbf{d}_3, \mathbf{d}_4\}$. *dist* is then computed from the Euclidean distance between $\mathbf{d}_4$ and $\mathbf{q}$. The expanded result (demarcated by the dark outer circle with radius

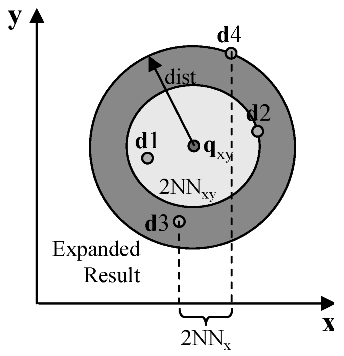


Fig. 5. Query processing.

dist) now contains the actual result $2\text{NN}_{xy} = \{\mathbf{d}_1, \mathbf{d}_2\}$ (demarcated by the light inner circle). After decrypting the y-coordinates, the user can rank $\mathbf{d}_1$ to $\mathbf{d}_4$ and locate the top two matches correctly.

Theoretically, the document server could disregard the suppressed columns in $\mathbf{V}$, and compute document similarity scores with $\mathbf{V}$'s r_2 remaining plaintext columns and $\mathbf{q}_{(r2)}$, as in LSI. In practice, doing so would yield very poor retrieval effectiveness: (a) If LSI is employed, $\mathbf{U}$, Σ and $\mathbf{V}$ would have been trimmed to the desired number of columns (typically only one or two hundreds [Dumais 1994]). A further loss of the suppressed columns would leave too few columns to effectively retain the important associations in $\mathbf{X}$, the term-document matrix. (b) To simplify our presentation, we have shown that trailing columns are suppressed. In fact, the suppressed columns could well be the leading ones or spread out uniformly, as we will see in Section 6. Disregarding the suppressed columns would thus introduce too much error to the similarity ranking.[4]

4.4 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 $k\text{NN}_{(r_1)}$. Following that, the user may proceed to click on the document links to download the content of those $k\text{NN}_{(r_1)}$ documents.

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[5] $E_T(m, k)$ that encrypts a message m with a key k such that $E_T(E_T(m, k_1), k_2) = E_T(E_T(m, k_2), k_1)$ [Bao et al. 2000]. The encrypted content of a document is paired with a randomly

[4]The impact of using too few factors on precision-recall performance is well documented, for example in Dumais [1994, 1995].

[5]A well-known commutative encryption function is ElGamal [ElGamal 1984]. It involves two modulo exponentiations. We clocked its encryption and decryption operations in the Crypto++ library (http://www.cryptopp.com) at just 11.6 msec and 36.6 msec respectively, even with 1024-bit long modulus. These CPU overheads are acceptable, taken in the context of other system costs like query processing and network transmission. The use of encryption also does not increase the index size, thus posing no space overhead.

generated, but unique document id to facilitate retrieval through an index (as explained in Section 6.1). Thus, each document $\mathbb{d}_j$ is stored in the form $\langle id_j, E_T(\mathbb{d}_j, k_c)\rangle$ where k_c is the encryption key for the corpus.

Referring to the text retrieval model in Figure 2, the user downloads the result documents $\{\langle id_1, E_T(\mathbb{d}_1, k_c)\rangle, \ldots\}$ in step 4. The encrypted content of each document $\mathbb{d}_j$ in the result is encrypted again with the user's key k_u – $E_T(E_T(\mathbb{d}_j, k_c), k_u) = E_T(E_T(\mathbb{d}_j, k_u), k_c)$ – and sent to the access manager in step 5. The access manager decrypts the documents with k_c, and the resulting $E_T(\mathbb{d}_j, k_u)$ are returned in step 6. The user then decrypts with k_u to recover $\mathbb{d}_j$.

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 $\mathbb{d}_j$ with a randomly generated document key k_j, then locks k_j with k_c. Thus, $\mathbb{d}_j$ is stored as $\langle id_j, E_C(\mathbb{d}_j, k_j), E_T(k_j, k_c)\rangle$ where E_C is a conventional encryption function like AES [2001]. The user can now send the encrypted document keys—$E_T(E_T(k_j, k_c), k_u)$—in step 5, and in step 6 the access manager returns $E_T(k_j, k_u)$. The user then decrypts with k_u to extract k_j, and decrypts $E_C(\mathbb{d}_j, k_j)$ to recover $\mathbb{d}_j$.

4.5 Detailed Indexing and Retrieval Protocols

The following indexing and retrieval protocols flesh out the proposed information retrieval scheme.

4.5.1 *Corpus Preparation.* The corpus preparation that the owner performs involves the following steps.

—Assign a randomly generated, but unique id_j to each document $\mathbb{d}_j$. Lock $\mathbb{d}_j$ into the form $\langle id_j, E_C(\mathbb{d}_j, k_j), E_T(k_j, k_c)\rangle$, where k_j is a randomly generated encryption key that is unique for each document, and k_c is the overall corpus key.

—Apply Singular Value Decomposition (SVD) to decompose the original term-document matrix $\mathbf{X}$ into a left singular matrix $\mathbf{U}$, a diagonal matrix of eigenvalues Σ, and a right singular matrix $\mathbf{V}$.

—$\mathbf{U}$ and Σ are encrypted with a random key k_{SVD}.

—For each document or column vector $\mathbf{d}_j$ in $\mathbf{V}^T$, create a triplet $\langle \mathbf{d}_{j(r_2)}, E_C(id_j|\delta\mathbf{d}_j, I_j), E_T(I_j, k_r)\rangle$ where I_j is a random index entry key, k_r is the overall index key, and $|$ is the concatenation operator.

After corpus preparation, the locked documents, the encrypted $\mathbf{U}$ and Σ, and the partially suppressed $\mathbf{V}$ are deposited on the document server. k_{SVD}, the corpus key k_c and index key k_r are kept by the access manager.

4.5.2 *Document Retrieval Protocol.* Consider steps 3 and 4 in Figure 2. Since the document server works on only the r_2 plaintext rows of $\mathbf{V}^T$, the query result will include some "false positives." Rather than returning the content of all those documents directly, the document server returns their complete document vectors (comprising plaintext and encrypted coordinates) first. Only after weeding out the false positives with the aid of the encrypted coordinates, does the user contact the document server again to download the actual document contents. Details are as follows.

The query result from the document server is a collection of entries of the form $\langle \mathbf{d}_{j(r_2)}, E_C(id_j|\delta\mathbf{d}_j, I_j), E_T(I_j, k_r)\rangle$. Using the twin-lock mechanism described in Section 4.4, the user sends the $E_T(E_T(I_j, k_r), k_u)$'s to the access manager, which decrypts with k_r and reverts with the $E_T(I_j, k_u)$'s. The user then obtains I_j by decrypting with k_u, recovers the plaintext id_j and $\delta\mathbf{d}_j$, and combines $\delta\mathbf{d}_j$ with $\mathbf{d}_{j(r_2)}$ to get the complete $\mathbf{d}_{j(r_1)}$. After ranking the documents with the complete $\mathbf{d}_{j(r_1)}$ and $\mathbf{q}_{(r_1)}$ vectors, the $k\text{NN}_{(r_1)}$ result documents can now be identified. Following that, the id_j's of the result documents in $k\text{NN}_{(r_1)}$ are presented to the document server, for it to return the locked document contents $\langle id_j, E_S(\mathbb{d}_j, k_j), E_T(k_j, k_c)\rangle$ as explained in Section 4.4.

The interactions between the user, document server and access manager during query processing are captured in the protocol in Figure 6. The protocol requires two rounds of exchanges between the user and the document server. This is advocated in Song et al. [2000] to prevent the document server from statistically correlating the index and the documents.

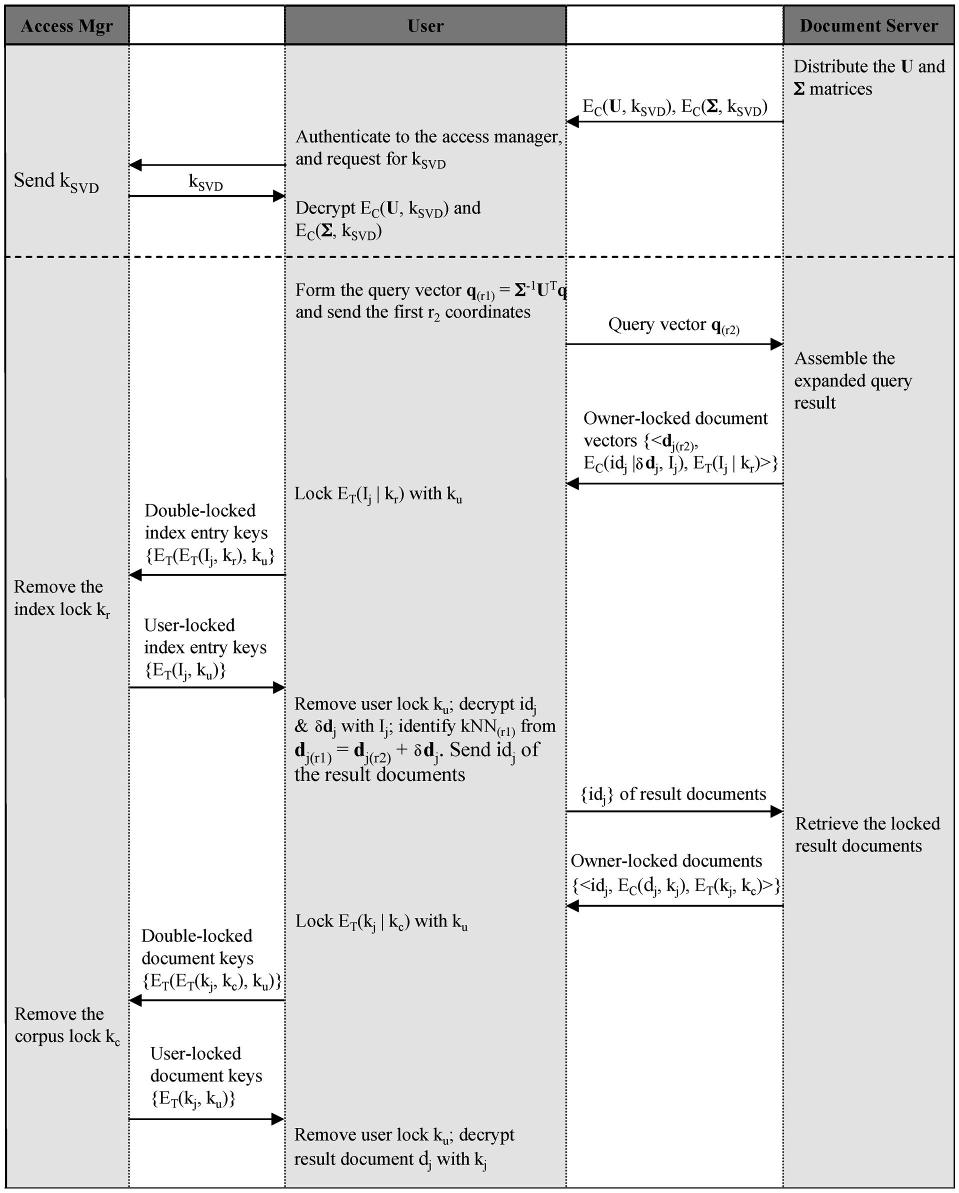
In addition, the index key k_r should be different from the corpus key k_c. This ensures that users cannot circumvent the access manager's usage accounting, by falsely presenting document encryption keys k_j's as index entry keys I_j's.

4.6 Scalability Considerations

As shown in the beginning of this section, our proposed scheme entails an SVD operation. A common concern is that SVD is computationally expensive, and may not be practical for very large term-document matrices. To overcome the problem, we could rely on more optimized SVD procedures [Fierro and Berry 2002] and Monte-Carlo algorithms that approximate the SVD computation [Drineas et al. 2006]. Specifically, we extract a sample $\mathbf{X}_{m\times p}$ of the term-document matrix $\mathbf{X}_{m\times n}$ where $p < n$, derive the singular matrices from the sample $\mathbf{X}_{m\times p} = \mathbf{U}_{m\times r_0} \cdot \Sigma_{r_0\times r_0} \cdot \mathbf{V}^T_{r_0\times p}$, then fold in the full matrix with $\mathbf{V}^T_{r_0\times n} = \Sigma^{-1}_{r_0\times r_0} \cdot \mathbf{U}^T_{r_0\times m} \cdot \mathbf{X}_{m\times n}$. We can now apply our scheme to $\mathbf{V}^T_{r_0\times n}$ as described earlier.

In this procedure, the rank r_0 has to be large enough for $\mathbf{X}_{m\times n}$ to be reconstructed accurately from $\mathbf{U}_{m\times r_0}$, $\Sigma_{r_0\times r_0}$ and $\mathbf{V}^T_{r_0\times n}$. By definition, $r_0 \leq min(m, p)$, so we should set the number of sampled documents p large enough that r_0 is limited only by m, the number of index terms in the corpus. In fact, $p = m$ is sufficient, as long as documents are not repeated in the sample. This is so because we only need to be able to reconstruct $\mathbf{X}$ accurately for one r_0 setting, which is a more relaxed requirement than in LSI where every truncation of the singular matrices must also minimize the reconstruction error. In the LSI case, the sample has to be representative of the entire term-document matrix $\mathbf{X}_{m\times n}$, thus necessitating the sample size p to be pegged to n.

Since our computation here is determined by m, and independent of the actual number of documents n, the SVD step is scalable to an arbitrarily large number of documents. In practice, many document collections contain around



Access Mgr
User
Document Server
Distribute the U and Σ matrices
E_C(U, k_SVD), E_C(Σ, k_SVD)
Authenticate to the access manager, and request for k_SVD
Send k_SVD
k_SVD
Decrypt E_C(U, k_SVD) and E_C(Σ, k_SVD)
Form the query vector q_(r1) = Σ^-1 U^T q and send the first r_2 coordinates
Query vector q_(r2)
Assemble the expanded query result
Owner-locked document vectors {<d_j(r2), E_C(id_j | δd_j, I_j), E_T(I_j | k_r)>}
Lock E_T(I_j | k_r) with k_u
Double-locked index entry keys {E_T(E_T(I_j, k_r), k_u}
Remove the index lock k_r
User-locked index entry keys {E_T(I_j, k_u)}
Remove user lock k_u; decrypt id_j & δd_j with I_j; identify kNN_(r1) from d_j(r1) = d_j(r2) + δd_j. Send id_j of the result documents
{id_j} of result documents
Retrieve the locked result documents
Owner-locked documents {<id_j, E_C(d_j, k_j), E_T(k_j, k_c)>}
Lock E_T(k_j | k_c) with k_u
Double-locked document keys {E_T(E_T(k_j, k_c), k_u)}
Remove the corpus lock k_c
User-locked document keys {E_T(k_j, k_u)}
Remove user lock k_u; decrypt result document d_j with k_j

Fig. 6. Document retrieval protocol.

50,000 index terms, and $r_0 = 1500$ is enough to reduce the reconstruction error to negligible levels (as shown in the experiments in Section 6). The resources needed for such an SVD operation are well within the capacity of a 64-bit computing server.

Besides SVD computation, user computation and storage associated with the **U** and Σ matrices may also affect the scalability of our solution. Suppose that $m = 50{,}000$ and $r_1 = 200$ (a sufficient setting according to experiment results in Dumais [1994] and Section 6.7). Representing each matrix element with a 4-byte floating point, $\mathbf{U}$ and Σ occupy roughly 40 Mbytes and 800 bytes respectively, without compression. We envisage that they will be installed as part of the client software, rather than being downloaded at runtime. To generate the reduced vector for a query $\mathbf{q}$, we first derive $\mathbf{U}^T \cdot \mathrm{q}$; assuming that $\mathbf{q}$ is a long query consisting of 20 terms, this step requires $200 \times 20 = 4{,}000$ multiplication operations. Next, we compute $\mathbf{q}_{(r1)} = \Sigma^{-1} \cdot (\mathbf{U}^T \cdot \mathrm{q})$ which incurs another 200 multiplications. In all, generating $\mathbf{q}_{(r1)}$ requires 4,200 multiplications, which can be completed in about 10 msec. As an optimization, we could compute and store $\Sigma^{-1} \cdot \mathbf{U}^T$ (instead of storing the two matrices separately), thus saving 800 bytes of storage and 200 multiplications per query. The requisite storage and computation overheads can be supported easily on modern hardware.

5. ANALYSIS OF PRIVACY SAFEGUARDS

Having presented our text retrieval scheme, we now analyze the privacy protection that it offers. We first examine the normal scenario where the document server and the access manager act independently, followed by scenarios where the two might collude.

5.1 Scenario A: Independent Document Server and Access Manager

Excluding the initial request for k_{SVD} to decrypt $\mathbf{U}$ and Σ, the protocol in Figure 6 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 $\mathbf{q}_{(r2)}$ and candidate result documents $\mathbf{d}_{j(r2)}$. Being vectors within a synthetic factor space that is derived through singular value decomposition, $\mathbf{q}_{(r2)}$ and $\mathbf{d}_{j(r2)}$ convey no useful meaning on their own, as illustrated in Figure 4. 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.

5.2 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, defined in Section 4.1. We denote the plaintext and suppressed representation of the query as $\mathbf{q}$ and $\mathbf{q}_{(r2)}$ 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

Term	Documents								
	$\mathbf{d}_1$	$\mathbf{d}_2$	$\mathbf{d}_3$	$\mathbf{d}_4$	$\mathbf{d}_5$	$\mathbf{d}_6$	$\mathbf{d}_7$	$\mathbf{d}_8$	$\mathbf{d}_9$
human	.286	.043	.344	.369	-.093	-.010	-.003	.005	.005
interface	.299	.094	.371	.384	-.034	.008	.002	-.006	-.014
computer	.158	.228	.216	.190	.163	-.032	-.014	.011	.069
user	.104	.405	.193	.113	.420	.035	.003	-.034	-.005
system	.353	.216	.452	.446	.072	-.000	-.001	-.002	.016
response	-.021	.490	.056	-.055	.571	.029	-.002	-.037	.019
time	-.021	.490	.056	-.055	.571	.029	-.002	-.037	.019
EPS	.311	.081	.385	.401	-.049	.019	.007	-.009	-.031
survey	.033	.284	.023	.006	.163	-.179	.013	.236	.481
trees	-.019	-.056	.030	.005	.053	.892	.802	.535	-.040
graph	.007	.053	-.022	-.003	-.055	.237	.425	.560	.519
minors	.017	.094	-.040	-.006	-.087	-.058	.217	.496	.678

Fig. 7. $\hat{\mathbf{X}} = \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^T_{(r2)}$ for the Sample Corpus, with $r_2 = 4$.

entries in the search result. Only the user, after deciphering the suppressed factors in the document vectors, can differentiate the two accurately.

Let $\mathbb{S}_c(a, b)$ denote a c-dimensional hypersphere with center a and radius b. $\mathbb{S}_{r_2}(\mathbf{q}_{(r_2)}, \max_j |\mathbf{d}_{j(r_2)} - \mathbf{q}_{(r_2)}|)$ is the smallest hypersphere that envelops the kNN$_{(r_2)}$ documents $\mathbf{d}_j$ from the initial search. Following the expanded search, the hypersphere is enlarged to a radius of *dist* (as defined in Formula (2)). If documents are uniformly distributed in the r_2-factor space, the increase in the number of documents is proportional to the volume increase of the hypersphere, thus the anonymity of the result for query $\mathbf{q}_{(r_2)}$ is roughly $(\frac{dist}{\max_j |\mathbf{d}_{j(r_2)} - \mathbf{q}_{(r_2)}|})^{r_2}$. As the uniform distribution assumption does not always hold in practice, however, we will measure the anonymity level empirically in Section 6.

5.3 Scenario C: Document Server Colludes with Users

Next, suppose that the document server colludes with a user to get hold of the decrypted left singular matrix $\mathbf{U}$ and Eigenvalues Σ. The document server can estimate the corpus by computing $\hat{\mathbf{X}} = \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^T_{(r2)}$, and the query with $\hat{\mathbf{q}} = \mathbf{U} \cdot \Sigma \cdot \mathbf{q}_{(r2)}$. The accuracy of these estimates are quantified by the fidelity metric, defined in Section 4.1. It can be shown that $||\mathbf{X}||_F = \sqrt{\sum_{i=1}^{r_1} \sigma_i^2}$, the sum of the r_1 highest Eigenvalues in Σ, and that $||\mathbf{X} - \hat{\mathbf{X}}||_F = \sqrt{\sum_{i=r_2+1}^{r_1} \sigma_i^2}$. Moreover, the fidelity can be calculated from the Eigenvalues in Σ:

$$fidelity = 1 - \sqrt{\frac{\sum_{i=r_2+1}^{r_1} \sigma_i^2}{\sum_{i=1}^{r_1} \sigma_i^2}}.$$

Continuing our running example from Figures 1 and 4, we show in Figure 7 the term-document matrix $\hat{\mathbf{X}}$ that the adversary is able to deduce with only the $r_2 = 4$ plaintext factors. The fidelity of the estimated matrix is around

0.56. Clearly, its document vectors contain many spurious terms, which add to the uncertainty in the adversary's deductions. In Section 6, we will study the fidelity levels that our scheme achieves with real corpora.

An alternative metric for quantifying the information leakage here might be derived from the notion of differential entropy in information theory, as suggested in Shen et al. [2007]. It would be interesting future work to investigate how entropy, which is defined for independent random variables, can be extended to the entire term-document matrix or the text corpus in general.

5.4 Scenario D: Dictionary Attacks

With the left singular matrix $\mathbf{U}$ and Eigenvalues Σ from a colluding user, the document server could also attempt a dictionary attack. There are two possible cases: (a) to find a query q' that has the exact term composition as the user query q, i.e., $q' = q$; and (b) to find a query q' that contains one or more terms in the user query q, i.e., $q' \subseteq q$.

(a) To find $q' = q$. Such an attack is straightforward with the Boolean model, as the result set for a query can be constructed from the result sets for the component terms. For example, the answer for "term$_1$ AND term$_2$" is the intersection of the component answers. With the vector space model that we adopt in this paper, however, a document that does not contain some of the search terms, but has a high similarity score relative to the remaining search terms, can still qualify for the query answer. To deduce the exact term composition of a user query, the document server would thus have to test all the powersets of the vocabulary terms, even discounting the possibility that the query may repeat some terms. As many realistic corpora contain 50,000 search terms or more, such an exhaustive enumeration is computationally feasible only for queries with two or at most three search terms. Admittedly, this might still pose a concern for Web search queries which are generally short. The countermeasure we can think of is to inform users who are concerned about privacy to formulate queries that are more specific and longer, and hence beyond the reach of brute force attack.

(b) To find $q' \subseteq q$. For longer queries like TREC topics,[6] a more feasible attack is to test whether a certain term in the vocabulary is included in a user query. This can be done by computing the search result for every term, then checking whether each term's result is similar to the user's query result. Specifically, the adversary enumerates the single-term queries $q_i = \{t_i\}$, where t_i is the i-th entry in the vocabulary of m search terms. For each q_i, the document server generates the result set $R(q_i, k)$ containing the top k matching documents. Now a user submits a query q, and gets a top-k result set $R(q, k)$. The adversary wants to find the probability that some suspected q_i is a subset of q, given the observed similarity between their respective result sets $sim(R(q_i, k), R(q, k))$, where $sim(R(q_i, k), R(q, k)) = 1$ if $R(q_i, k)$ and $R(q, k)$ are judged to be similar

[6]Text REtrieval Conference. http://trec.nist.gov/.

and 0 otherwise. By Bayes rule,

$$\begin{aligned} P(q_i \subseteq q|sim(R(q_i,k), R(q,k)) = 1) \\ = P(sim(R(q_i,k), R(q,k)) = 1|q_i \subseteq q)P(q_i \subseteq q) \; / \\ \big[P(sim(R(q_i,k), R(q,k)) = 1|q_i \subseteq q)P(q_i \subseteq q) + \\ P(sim(R(q_i,k), R(q,k)) = 1|q_i \nsubseteq q)P(q_i \nsubseteq q)\big]. \end{aligned} \quad (3)$$

In this equation, $P(q_i \subseteq q)$ can be estimated from the relative frequency of term t_i in the text corpus or historical queries, while $P(q_i \nsubseteq q)$ is roughly $\frac{m-|q|}{m}$. $P(sim(R(q_i,k), R(q,k)) = 1|q_i \subseteq q)$ is approximated through the following procedure: Treating $R(q_i,k)$ and $R(q,k)$ as n-dimensional vectors with coordinate i set to 1 if document d_i is in the result and 0 otherwise, we compute the cosine similarity[7] between $R(q_i,k)$ and $R(q,k)$. Since both vectors contain only $\{0, 1\}$ values, the cosine similarity ranges between 0 when the result sets share no common documents, and 1 when the two results are identical. We therefore use the cosine similarity as an estimator for $P(sim(R(q_i,k), R(q,k)) = 1|q_i \subseteq q)$. The rationale is that, suppose we have a classifier for judging whether $R(q_i,k)$ and $R(q,k)$ are similar, we would expect it to have a higher probability of pronouncing the result sets to be similar when the cosine similarity is nearer 1, and a lower probability of a positive judgment when the cosine similarity is nearer 0. Similarly, $P(sim(R(q_i,k), R(q,k)) = 1|q_i \nsubseteq q)$ is estimated by the average cosine similarity between $R(q,k)$ and the results of nonquery terms.

As we will demonstrate empirically in Section 6.8, $P(sim(R(q_i,k), R(q,k)) = 1|q_i \subseteq q)$ is not significantly higher than $P(sim(R(q_i,k), R(q,k)) = 1|q_i \nsubseteq q)$. Given the sheer magnitude of m (the number of vocabulary terms), we expect $P(q_i \subseteq q) \ll P(q_i \nsubseteq q)$. Therefore, the denominator in Equation (3) dwarfs the numerator, implying a low certainty in any deduction on what search terms might be in a user query. The exception is where there are a few disproportionately frequent terms for which $P(q_i \subseteq q) \not\ll P(q_i \nsubseteq q)$. However, such terms are likely to be discarded as stopwords (that have little discriminatory value for retrieval purposes [Baeza-Yates and Neto 1999]) by the document server, and they do not disclose specific information about the query and user intention in any case.

The dictionary attack can be extended beyond single-term queries. In particular, an adversary could isolate a very small subset of interesting terms, then enumerate q' over the powerset of those terms to test for $q' \subseteq q$. If the user query q happens to contain only the isolated terms, the adversary would find a q' that produces an identical result as q. If q contains any term that is outside of the isolated subset so that $q' \subsetneq q$, however, a similar analysis and conclusion as for single-term queries apply.[8] Therefore the attack is effective only when the user queries are limited strictly to a small yet informative set of

[7]The cosine similarity between two n-dimensional vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as $\frac{\mathbf{x}\cdot\mathbf{y}}{\sqrt{|\mathbf{x}|}\sqrt{|\mathbf{y}|}}$.

[8]Lengthening the probe query by one term will increase the number of possible q' by a factor of m, while reducing $P(q' \subseteq q)$ by a factor of $|q|/m$ where $|q|$ is the number of unique query terms. Since $P(q' \nsubseteq q) = 1 - P(q' \subseteq q)$, a longer q' will cause the denominator to overwhelm the numerator in Equation 3, and the probability of deducing the search terms to approach zero quickly.

search terms, and it can be defeated easily once users add extra relevant terms to make their queries longer and more specific.

5.5 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 Eigenvalues 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.

6. EXPERIMENTS

In the previous section, we have identified the privacy risks that ensue if the document server manages to obtain extra information by colluding with other users or the access manager. We have also shown how the *fidelity* and *anonymity* metrics are used to quantify those privacy risks. We now present an empirical study of these metrics with a prototype system that we implemented.

6.1 Description of Prototype

Our implementation first uses the Lucene search engine to build an index from the corpus. Next, we 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 according to the formulation in Section 2.2. The document vectors are normalized, before using the SVD routine in matlab to generate the **U**, Σ 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 kNNs (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, we construct an R-tree index [Guttman 1984] over the document vectors in **V**.

6.1.1 *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.

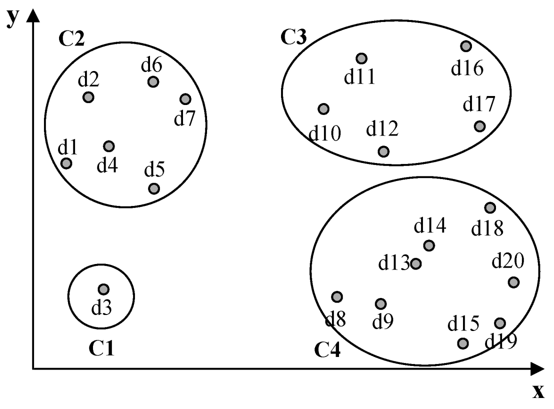


Fig. 8. Clustering the document vectors.

To illustrate, suppose that the matrix **V** generated from SVD contains 20 document vectors (d1 to d20), as plotted in Figure 8. The clustering algorithm organizes them into 4 clusters (C1 to C4).

6.1.2 *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 [Robinson 1981]. Specifically, we section a large partition along the first of the r_2 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.

6.1.3 *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 *bulk-loaded* [Berchtold et al. 1998] 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. The MBRs for the 12 partitions and the resulting R-tree are illustrated in Figures 10 and 11, respectively.

Compared to loading the documents directly into the R-tree, our procedure of pre-organizing the documents into partitions reduces the extent of the tree nodes, as well as the overlap between nodes at the same level of the R-tree. The procedure also provides a basis to set the R-tree's dimensionality. The price

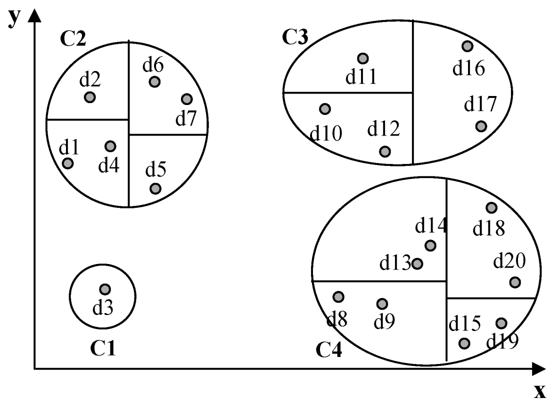


Fig. 9. Clustering & partitioning.

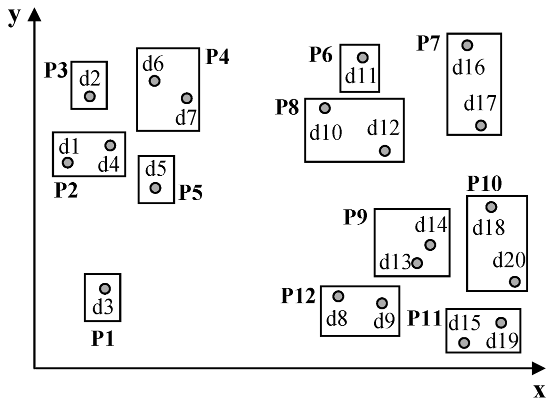


Fig. 10. Defining partition MBR.

that we pay is the overhead of generating the partitions. This should not be a big deterrent though, because the document servers re-generate their indices periodically rather than update them on-the-fly as documents are added. Moreover, our experience shows that clustering and partitioning take only a small fraction of the time consumed by SVD.

6.1.4 *Retrieval through the Index.* Suppose that the R-tree $\mathbb{R}$ constructed above indexes the first r_3 of the r_2 plaintext factors in $\mathbf{V}$. We need to apply the procedure in Section 4.3 twice to generate the query result

—Phase 1: Locate the k documents that are nearest the query in the r_3-dimensional space, $k\mathrm{NN}_{(r_3)}$. Compute the maximum distance $dist_1$ between these $k\mathrm{NN}_{(r_3)}$ documents and the query in the r_2-dimensional space.

—Phase 2: Locate all the documents that are up to a distance of $dist_1$ 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 $k\mathrm{NN}_{(r_2)}$. Compute the maximum distance $dist_2$ between these $k\mathrm{NN}_{(r_2)}$ documents and the query in the r_1-dimensional space.

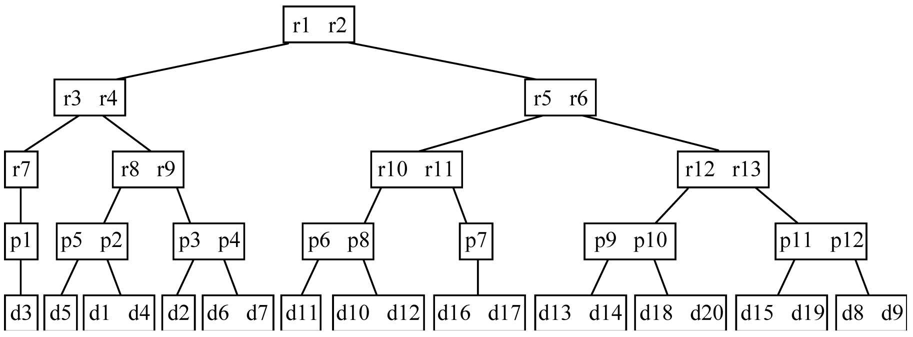


Fig. 11. An R-Tree for the running example.

—Phase 3: Retrieve all the documents that reside up to a distance of $dist_2$ from the query in the r_3-dimensional space, using the R-tree. From these documents, prune away those that are beyond $dist_2$ from the query in the r_2-dimensional space. The remaining documents will include $k\mathrm{NN}_{(r_1)}$, the actual result documents.

The lemma in Section 4.3 can be extended easily to prove that this procedure is guaranteed to produce a superset of the actual result documents. An efficient algorithm for searching the R-tree is given in Hjaltason and Samet [1999].

6.2 Experiment Set-Up

Our text retrieval scheme achieves privacy protection by leaving a subset of the document coordinates in plaintext, while encrypting the remaining coordinates. Obviously, the achieved fidelity and anonymity levels depend on the $mask\% = 1 - r_2/r_1$, the percentage of coordinates that are encrypted. For simplicity, in Section 4 we have demonstrated that the plaintext coordinates are in the top rows in the suppressed document matrix $\mathbf{V}^T$, that is, those that correspond to the most significant Eigenvalues. Of course, there are other ways to select the plaintext versus encrypted coordinates. We will evaluate the following masking schemes.

—*Prefix* masking. Among the r_1 rows retained after SVD (and LSI if applicable), leave the r_2 corresponding to the smallest Eigenvalues in plain and encrypt the other rows.
—*Suffix* masking. Leave the rows in $\mathbf{V}^T$ corresponding to the r_2 largest Eigenvalues in plaintext, and encrypt the remaining coordinates.
—*Spaced* masking. Spread out the encrypted coordinates across the r_1 rows in equal intervals.

We 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, as defined in Lewis et al. [2004]. The characteristics of the resulting dataset are summarized in Table II.

Table II. Characteristics of the Datasets

Parameter	WSJ	RCV1
n: # of documents in the dataset	10,123	23,149
m: # of terms in the dataset	46,493	47,236
r_1: Default # of coordinates kept by LSI	1,500	1,500
mask%: Ratio of r_1 coordinates that are encrypted ($= 1 - r_2/r_1$)	–	–

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

—$precision = \frac{\text{\# of relevant documents in the search result}}{\text{total \# of documents in the search result}}$,

—$recall = \frac{\text{\# of relevant documents in the search result}}{\text{total \# of relevant documents}}$.

We also want to measure the privacy risk in circumstances where the document server manages to acquire extra information by colluding with the access manager or other users. As defined in Section 4.1, the privacy metrics are the following.

—The *fidelity* of the approximated corpus and user queries that could be reverse-engineered from the suppressed $\mathbf{V}_{(r_2)}$ and $\mathbf{q}_{(r_2)}$. Fidelity can be calculated directly from the Eigenvalues of the term-document matrix $\mathbf{X}$ as explained earlier.

—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.

The efficiency of the query result—the fraction of useful data in the expanded result that is returned to the user—is inversely proportional to the anonymity metric. Moreover, all the other steps in the protocol in Figure 6 are pegged to the number and size of the documents that the user downloads. As such, we will focus the experiment results on the four metrics above.

6.3 Parameter Calibration

The first configuration parameter in our solution is r_1, the number of retained Eigenvalues. Dumais [1995] showed that the precision-recall of Latent Semantic Indexing (LSI) typically peaks in the range $200 < r_1 < 350$, and converges to the vector space model as r_1 is raised further. However, a more recent study [Husbands et al. 2001] reported that while truncating the singular matrices (i.e., $\mathbf{U}$, Σ and $\mathbf{V}$) as done in LSI could improve retrieval effectiveness for a small homogeneous corpus, it is not suitable for large heterogeneous text collections. Another criticism is that LSI is designed for normally distributed data, but the term-document matrix (even if weighted) from a text corpus may not be normally distributed [Rosario 2000]. Since LSI is not the contribution of our article, we start our experiments with a high r_1 of 1500 to approximate the vector space model as closely as our computing resources would allow. We will discuss the effect of lowering r_1 in a later experiment.

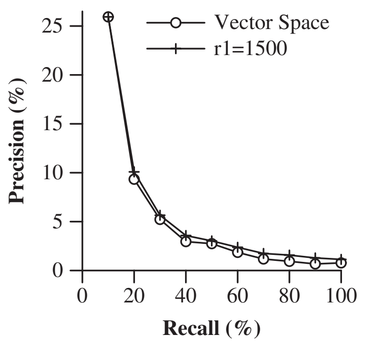


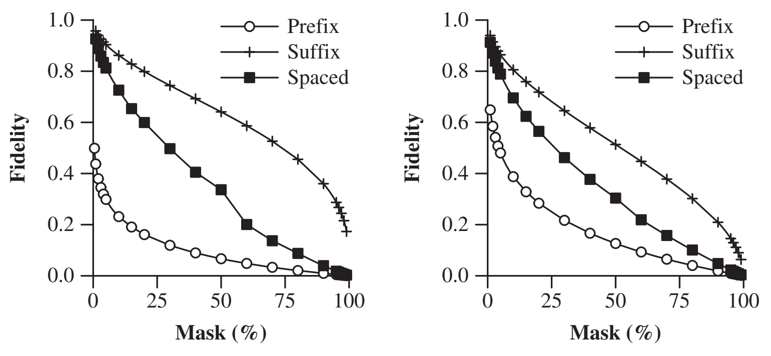
Fig. 12. Precision-Recall.

To confirm that our r_1 setting behaves as expected, we carry out an experiment with the TREC-2 and TREC-3 adhoc queries (topics 101 to 200) on the WSJ corpus. The queries contain between two and 20 terms each, and provide realistic term compositions for testing our solution. After eliminating those topics for which there are less than 10 relevant documents, the precisions observed at various standard recall levels are averaged. Our results, summarized in Figure 12, are worse than those reported in the TREC proceedings [Dumais 1994, 1995]; that is primarily because we neither tuned our scoring function nor supported phrases. Nevertheless, the experiment serves its purpose, in confirming that the $r_1 = 1500$ setting produces almost identical precision-recall performance as the vector space model.

6.4 Fidelity versus Anonymity

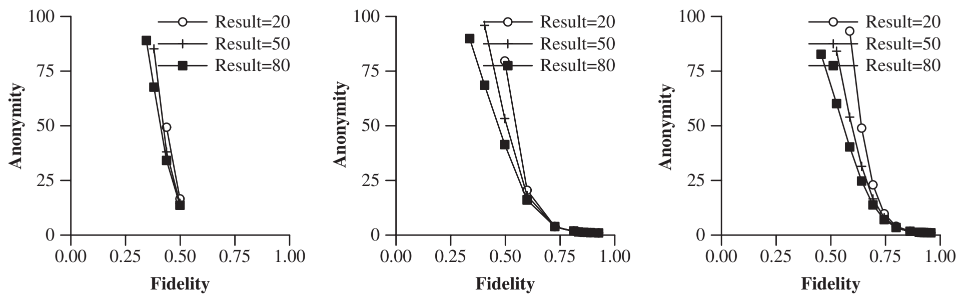
Figure 13 plots the fidelity levels for the three mask selections. Since *Prefix* suppresses the coordinates that correspond to the most significant Eigenvalues, it naturally leads to the lowest fidelity levels; indeed, by masking just 1% of the coordinates, the fidelity is reduced to 0.44 for WSJ, and 0.65 for RCV1. *Suffix* is the opposite of *Prefix* in suppressing the least significant coordinates. Setting *mask%* to 1% only brings the fidelity down to around 0.95. However, by raising *mask%*, even *Suffix* is able to limit the fidelity to very low levels. The fidelity of *Spaced* is in between *Prefix* and *Suffix* by design. This indicates that any suspected privacy risk resulting from a collusion between the document server and the access manager can be managed very effectively with any of the masking schemes.

Next, we examine the anonymity levels. The fidelity versus anonymity curves, for result sizes of 20, 50 and 80 documents respectively, are shown in Figures 14 and 15. The figures confirm that our scheme can concurrently achieve low fidelity and high anonymity by simply raising *mask%*. However, the performance overhead is proportional to the anonymity level, as the false-positives that provide anonymity also need to be sent to and processed by the user. Therefore, while we want to limit any privacy leaks in the event of collusion, practically we should not allow the anonymity level to rise out of control especially if the probability of collusion is low.



(a) WSJ (b) RCV1

Fig. 13. Fidelity.



(a) Prefix (b) Spaced (c) Suffix

Fig. 14. WSJ: Fidelity versus anonymity.

Suppose we desire an anonymity target of 10, that is, every legitimate result document is mixed with 9 false-positives. Figure 14 shows that *Prefix*, *Suffix* and *Spaced* achieve that target at fidelity levels of around 0.7 for WSJ, while Figure 15 indicates fidelity levels between 0.6 and 0.7 for RCV1. Since all three masking schemes deliver similar fidelity versus anonymity protections, the choice can be determined by their relative ease of configuration, and runtime stability as we will examine next.

6.5 Configuration Granularity

After indexing the corpus, the data owner has to decide on the *mask%* for the suppressed representation to be used by the document server, without the benefit of examining the runtime search queries. We expect the setting to be guided by the target fidelity and anonymity levels, which can be computed from the corpus as explained in Section 6.2. Since the target anonymity level should not be too high on account of the concomitant overheads, it is desirable for the useful range of anonymity levels to be achieved over a wide spread of *mask%*. This allows the target fidelity-anonymity to be tuned more finely through the *mask%* setting at corpus preparation.

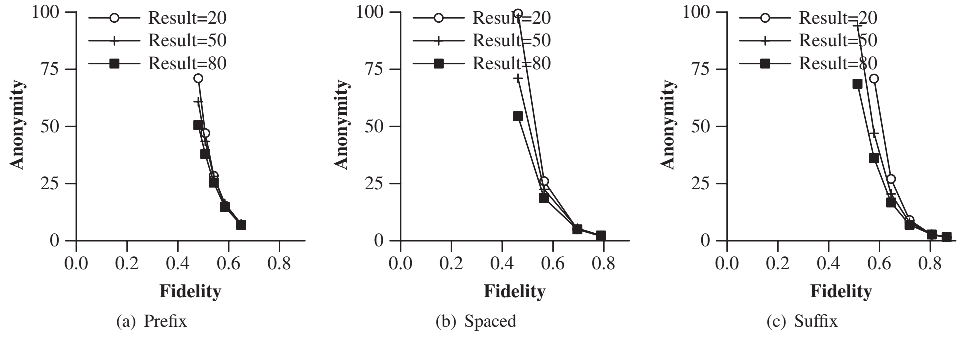


Fig. 15. RCV1: Fidelity versus anonymity.

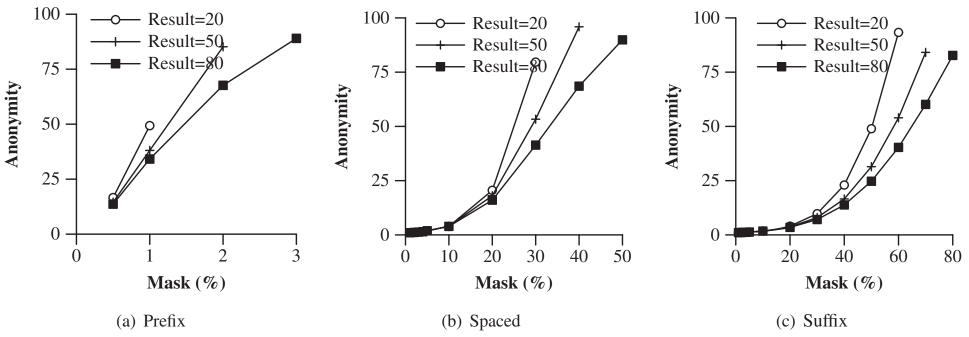


Fig. 16. WSJ: Sensitivity to mask%.

Figures 16 and 17 plot the anonymity as a function of *mask%* for the two corpora. Taking (1, 10] as the useful anonymity range, the *mask%* for WSJ is $<0.5\%$, $\leq 15\%$, and $\leq 30\%$ for *Prefix*, *Spaced* and *Suffix*, respectively. In the case of RCV1, the corresponding *mask%* is 1% for *Prefix*, $\leq$ 15% for *Spaced*, and $\leq 20\%$ for *Suffix*. This indicates that *Prefix* masking likely does not provide sufficient configuration flexibility unlike *Suffix* and *Spaced*, especially for smaller corpora with low term-document matrix ranks r_0.

6.6 Runtime Stability

Document servers frequently allow users to request for different number of result documents for their search queries. Therefore it is desirable for our text retrieval scheme to be able to maintain roughly the same anonymity levels across a wide range of result sizes, without having to tweak the *mask%*. The reason is that the *mask%* setting affects the document vectors in $\mathbf{V}$, and cannot be altered dynamically after the initial configuration.

Figures 18 and 19 plot the anonymity level versus result size for WSJ and RCV1. The experiment shows that all three mask selections are able to achieve different anonymity ranges for query result sizes of 5 and beyond, by setting *mask%* appropriately. If the search query retrieves only the top one or two matching documents, however, the anonymity levels are consistently low. The

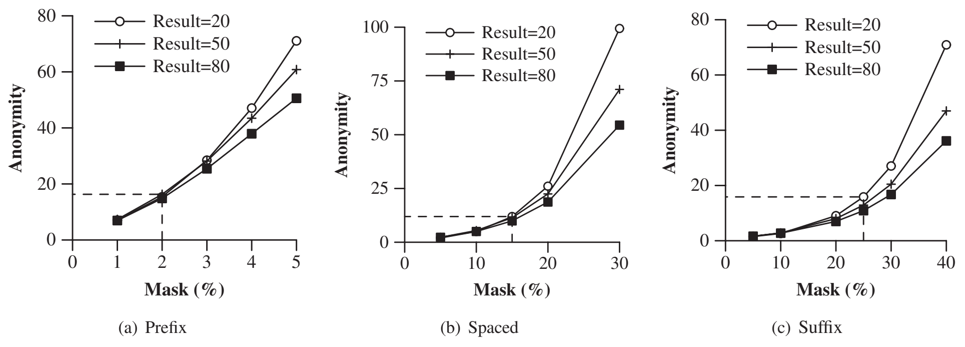


Fig. 17. RCV1: Sensitivity to mask%.

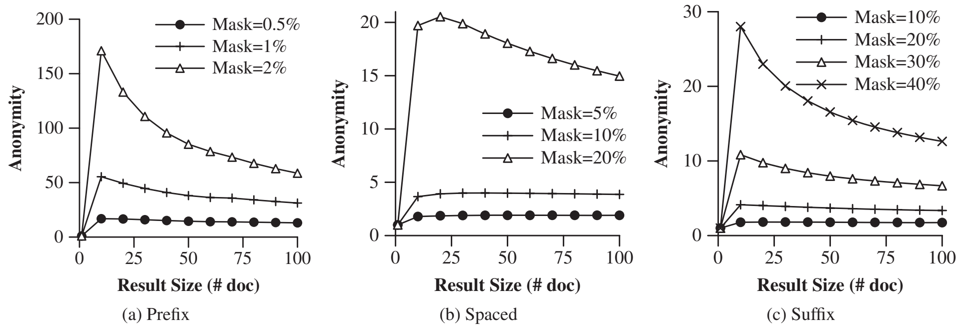


Fig. 18. WSJ: Sensitivity to search result size.

reason is that, with just one or two result documents, the search radius does not extend far during the expanded search (Section 4.3) to bring in enough of the neighboring documents. This is probably not a concern for desktop search applications, where the user expects to browse through several result documents. In contrast, applications running on mobile devices are likely to display at most a handful of result documents; to maintain the target anonymity level, the user device will need to request for a slightly larger result size (say, top-5), then trim the result locally to fit the display. Finally, we note again that the useful range of *mask%* for *Prefix* is very restrictive.

6.7 Latent Semantic Indexing

Now we consider the situation where, instead of retrieving text with similar term compositions, the document server is configured to perform concept-based retrieval with latent semantic indexing (LSI) [Deerwester et al. 1990]. Following the recommendation in Dumais [1995], we vary r_1 within the range of [200, 350]. Figure 20 plots the precision at various recall levels along with the fidelity measures, for the WSJ corpus. The results confirm that there is lit-tle threat that the document server could reconstruct the term composition of the documents or queries, even if it gets hold of the singular matrix **U** and Eigenvalues Σ through collusion.

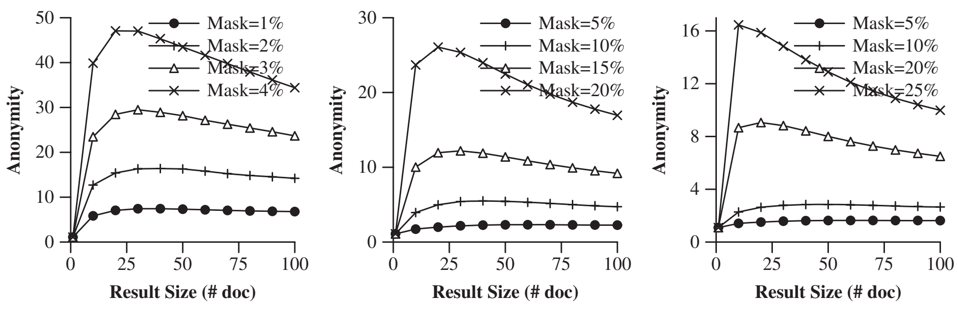


Fig. 19. RCV1: Sensitivity to search result size.

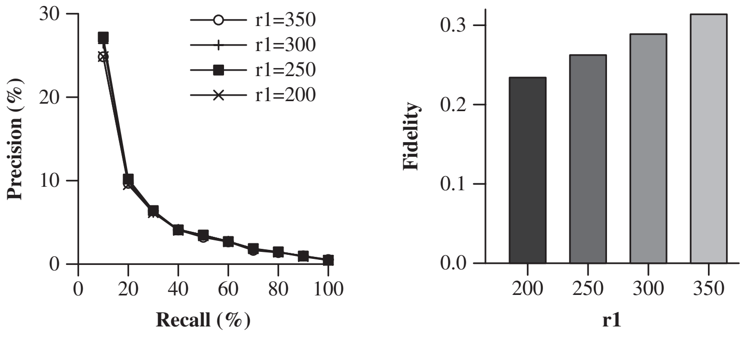


Fig. 20. WSJ: Latent semantic indexing.

The issue of anonymity is more subtle here. In the vector space model, documents are ranked by the similarity of their *term composition* to the user query. If the document server knows the plaintext of some documents, and those documents are ranked highly in the result of a search query whose plaintext is hidden from the server, it can still deduce the query terms from the known documents. Thereafter, it is up to the observer at the document server to interpret the user intention behind that query. To prevent that, it is necessary to induce uncertainty by adding false matches into the search result, as performed by our query processing algorithm and as measured by the anonymity metric.

With LSI, documents are ranked by their proximity to the query in the truncated factor space. Ideally, terms and documents that relate to common concepts form clusters in the factor space. In practice, many terms have multiple meanings, and documents often describe more than one topic. This implies that any known documents that rank highly in the result of a search query likely do not share many common terms with it, so the observer at the document server is no longer able to deduce the query terms. Instead, the observer has to hope that the known documents describe the same concept(s) as the query; in other words, the "interpretation" is taken away from the observer and embedded in LSI. This "interpretation" is rather imprecise as indicated in Figure 20(a). For example, LSI achieves a precision of 10% at 20% recall, which means that on average there are 9 irrelevant documents for every relevant document in the search results. Consequently, the irrelevant entries in the search result naturally provide anonymity protection to the relevant documents; there is no need to inject further false positives into the search result as we do for the vector space model.

Hence, we conclude that privacy of user queries is still maintained under LSI.

6.8 Dictionary Attack

In Section 5.4, we formulated the problem of dictionary attack. We now substantiate the analysis there by examining the similarity between the result sets of different queries. Using the WSJ corpus, we generate queries q with varying number of search terms, and measure the "qTerm" similarity (i.e., the cosine similarity between the result sets of q and q_i, where q_i is a query that comprises one of the terms in q). We also measure the "Rand" similarity (i.e., the cosine similarity between the result set of q and that of random queries, each of which is made up of one of the nonquery terms). The results for Suffix masking with 10% mask ratio are shown in Figure 21; each result point in the figure is averaged over 1000 queries.

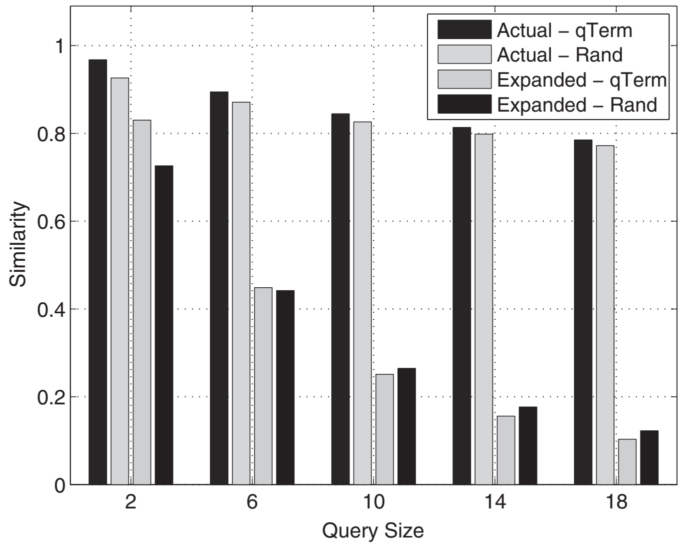
Figure 21(a) lists the cosine similarities for queries comprising different number of search terms, with the top 20 matching documents in each query result. For each query size, we give the "qTerm" versus "Rand" similarities for the actual query results, as well as for the anonymized results. We observe that our solution significantly lowers the similarity between query results, especially for longer queries, as a consequence of the false matches that our solution introduces into the expanded results to provide anonymity. More importantly, the gap between "qTerm" and "Rand" similarities, whether for the actual results or the anonymized results, is not wide enough for the query terms to dominate the nonquery terms in Equation (3), even for short queries containing only two search terms. This observation persists across different result sizes, as seen in Figure 21(b), which plots the similarity against result size with the query size fixed at 10 terms.

We have also experimented with Prefix and Spaced masking, and other masking ratios. The behaviors are similar to those observed in Figure 21. The experiment confirms our earlier analysis that the denominator in Equation (3) is much larger than the numerator, and that dictionary attacks do not yield high-confidence deductions about the search terms in a user query.

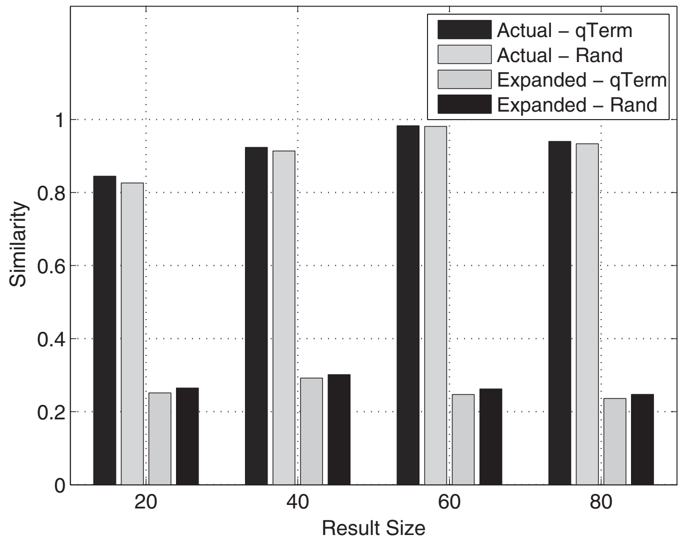
6.9 Discussion

The key observations from the experiment results are: (a) Our proposed text retrieval scheme enables the data owner to limit any privacy leaks in the event that the document server colludes with other parties to gain extra information, by setting the *mask%* appropriately at corpus preparation to achieve a wide range of target fidelity and anonymity levels. (b) Among the masking schemes, *Suffix* has the widest range of useful *mask%*'s, enabling it to work with even small corpora with as few as a hundred concept terms.

More generally, this article demonstrates that privacy-preserving, similarity-based text retrieval can be achieved by suppressing selected features



(a) Similarity versus Query Size



(b) Similarity versus Result Size

Fig. 21. WSJ: Dictionary attack.

in the document/query representation. The suppressed representation forces false positives into the search results that the document server generates in order to provide anonymity for the user queries, without wrongly dropping legitimate result documents. Since our proposed scheme is built on SVD, it fits naturally with LSI for which the retrieval performance has been studied extensively [Dumais 1994, 1995]. Where LSI is not desirable, setting $r_1 = r_0$

would revert to the vector space model, and our scheme would still work. Our technique could conceivably be adapted for alternatives to SVD, such as probabilistic LSI [Hofmann 1999] and semidiscrete matrix decomposition [Kolda and O'Leary 1998]; the challenge is to derive the corresponding anonymity and fidelity formulae.

7. 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.

Our work can continue in several interesting directions. First, while many practical search engines are built on the vector space model, they often employ complementary mechanisms to improve retrieval effectiveness. Notably, many Web search engines exploit the metadata of documents and the hyperlink structure between documents to boost the ranking of documents that are likely to be authoritative (e.g., Brin and Page [1998], Kleinberg [1999]). We intend to investigate how the metadata and hyperlinks can be protected, without crippling the associated ranking functions. Second, there are alternative text retrieval mechanisms to the vector space model, such as probabilistic relevance [Robertson and Jones 1976; van Rijsbergen 1979; Fuhr 1992] and probabilistic inference [van Rijsbergen 1986; Salton 1991]. Designing privacy protection schemes for these models would be a challenging undertaking.

ACKNOWLEDGEMENTS

We thank the reviewers for their constructive feedback.

REFERENCES

AES. 2001. Advanced Encryption Standard. *National Institute of Science and Technology*, FIPS 197.

Agrawal, R., Kiernan, J., Srikant, R., and Xu, Y. 2002. Hippocratic databases. In *Proceedings of the Very Large Data Bases*. 143–154.

Agrawal, R., Kiernan, J., Srikant, R., and Xu, Y. 2004a. Order preserving encryption for numeric data. In *Proceedings of the ACM International Conference on Management of Data*. 563–574.

Agrawal, R., Srikant, R., and Thomas, D. 2004b. Privacy preserving OLAP. In *Proceedings of the ACM International Conference on Management of Data*. 251–262.

Anderson, R., Needham, R., and Shamir, A. 1998. The steganographic file system. In *Proceedings of the International Workshop on Information Hiding (IWIH)*.

Baeza-Yates, R. and Neto, B. R. 1999. *Modern Information Retrieval*. Addison Wesley.

Bao, F., Deng, R. H., and Feng, P. 2000. An efficient and practical scheme for privacy protection in e-commerce of digital goods. In *Proceedings of the 3rd International Conference on Information Security and Cryptology, 2015*, 162–170.

Barbaro, M. and Zeller, T. 2006. A face is exposed for AOL searcher No. 4417749. *The New York Times*. http://www.nytimes.com/2006/08/09/technology/09aol.html.

Bawa, M., Bayardo, R. J., and Agrawal, R. 2003. Privacy-preserving Indexing of Documents on the Network. In *Proceedings of the International Conference on Very Large Data Bases.*

Berchtold, S., Bohm, C., and Kriegel, H.-P. 1998. Improving the query performance of high-dimensional index structures by bulk-load operations. In *Proceedings of the 6th International Conference on Extending Database Technology*. 216–230.

Bethencourt, J., Song, D., and Waters, B. 2006. New constructions and practical applications for private stream searching (extended abstract). In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'06)*. 132–139.

Bloom, B. 1970. Space/time trade-offs in hash coding with allowable errors. *Comm. ACM 13,* 7, 422–426.

Bookstein, A., Klein, S. T., and Ziff, D. 1992. A systematic appproach to compressing a full text retrieval system. *Inform. Process. Manage. 28,* 6, 795–806.

Brin, S. and Page, L. 1998. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst. 30,* 1–7, 107–117.

Chaum, D. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commu. ACM 24,* 2, 84–90.

Chor, B., Goldreich, O., Kushilevitz, E., and Sudan, M. 1995. Private information retrieval. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*. 41–50.

Damiani, E., Vimercati, S. D. C., Jajodia, S., Paraboschi, S., and Samarati, P. 2003. Balancing confidentiality and efficiency in untrusted relational DBMSS. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*. 93–102.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. 1990. Indexing by latent semantic analysis. *J. Am. Soc. Inform. Sci. 41,* 6, 391–407.

Dingledine, R., Mathewson, N., and Syverson, P. 2004. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*. 303–320.

Domingo-Ferrer, J., Sebé, F., and Castell-Roca, J. 2004. On the security of noise addition for privacy in statistical databases. In *Privacy in Statistical Databases*, 149–161.

Domingo-Ferrer, J., Viejo, A., Sebé, F., and González-Nicolás, U. 2008. Privacy homomorphisms for social networks with private relationships. *Comput. Netw. 52*, 3007–3016.

Drineas, P., Kannan, R., and Mahoney, M. W. 2006. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM J. Comput. 36,* 1, 158–183.

Dumais, S. T. 1994. Latent Semantic Indexing (LSI) and TREC-2. In *Proceedings of the 2nd Text REtrieval Conference (TREC2).* D. Harman, Ed. National Institute of Standards and Technology Special Publication, 105–116.

Dumais, S. T. 1995. Latent semantic indexing (LSI): TREC-3 report, D. Harman, ed. In *Proceedings of the 3rd Text REtrieval Conference (TREC3)*. National Institute of Standards and Technology Special Publication, 219–230.

Duncan, G. T., Fienberg, S. E., Krishnan, R., Padman, R., and Roehrig, S. F. 2001. Disclosure limitation methods and information loss for tabular data. In *Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies*, P. Doyle, J. Lane, J. Theeuwes, and L. Zayatz, Eds. Elsevier, 135–166.

ElGamal, T. 1984. A public-key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of the Crypto*. 10–18.

Fierro, R. and Berry, M. 2002. Efficient computation of the Riemannian SVD in TLS problems in information retrieval. In *Total Least Squares and Errors-In-Variables Modeling: Analysis, Algorithms, and Applications*, S. van Huffel and P. Lemmerling, Eds. Kluwer Academic Publishers, 349–360.

Freedman, M. J., Ishai, Y., Pinkas, B., and Reingold, O. 2005. Keyword search and oblivious pseudorandom functions. In *Proceedings of the 2nd Theory of Cryptography Conference*.

Fuhr, N. 1992. Probabilistic models in information retrieval. *Comput. J. 35,* 3, 243–255.

Goh, E. 2003. Secure indexes. Cryptology ePnnt Archive, Report 2003/216.

Goldschlag, D., Reed, M., and Syverson, P. 1999. Onion routing. *Comm. ACM 42,* 2, 39–41.

Gopal, R., Garfinkel, R., and Goes, P. 2002. Confidentiality via camouflage: The CVC approach to disclosure limitation when answering queries to databases. *Oper. Reser. 50,* 3, 501–516.

Gruteser, M. and Grunwald, D. 2003. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys'03)*, 31–42.

Guttman, A. 1984. R-Trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM International Conference on Management of Data*. 47–57.

Hacigumus, H., Iyer, B., Li, C., and Mehrotra, S. 2002. Executing SQL over encrypted data in the database service provider model. In *Proceedings of the ACM International Conference on Management of Data*. 216–227.

Hansell, S. 2006. Marketers trace paths users leave on internet. *The New York Times*. http://www.nytimes.com/2006/08/15/technology/15search.html.

He, X., Cai, D., Liu, H., and Ma, W.-Y. 2004. Locality preserving indexing for document representation. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 96–103.

Hjaltason, G. R. and Samet, H. 1999. Distance browsing in spatial databases. *ACM Trans. on Datab. Syst. 24,* 2, 265–318.

Hofmann, T. 1999. Probabilistic latent semantic indexing. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 50–57.

Husbands, P., Simon, H., and Ding, C. H. Q. 2001. On the use of the singular value decomposition for text retrieval. In *Proceedings of the SIAM Conference on Computational Information Retrieval*. 145–156.

Jajodia, S., Samarati, P., Sapino, M. L., and Subrahmanian, V. S. 2001. Flexible support for multiple access control policies. *ACM Trans. Datab. Syst. 26,* 2, 214–260.

Jiang, W., Murugesan, M., Clifton, C., and Si, L. 2008. Similar document detection with limited information disclosure. In *Proceedings of the IEEE International Conference on Data Engineering*. 735–743.

Kadane, J. B., Krishnan, R., and Shmueli, G. 2006. A data disclosure policy for count data based on the COM-Poisson distribution. *Manag. Sci. 52,* 10, 1610–1617.

Kalnis, P., Ghinita, G., Mouratidis, K., and Papadias, D. 2007. Preventing location-based identity inference in anonymous spatial queries. *IEEE Trans. Knowl. Data Engin. 19,* 12, 1719–1733.

Kim, J. 1986. A method for limiting disclosure in microdata based on random noise and transformation. In *Section on Survey Research Methods*, American Statistical Association, 303–308.

Klein, S. T., Bookstein, A., and Deerwester, S. C. 1989. Storing text retrieval systems on CD-ROM: Compression and encryption considerations. *ACM Trans. Inform. Syst. 7,* 3, 230–245.

Kleinberg, J. M. 1999. Authoritative sources in a hyperlinked environment. *J. ACM 46,* 5, 604–632.

Kolda, T. G. and O'Leary, D. P. 1998. A semidiscrete matrix decomposition for latent semantic indexing information retrieval. *ACM Trans. Inform. Syst. 16,* 4, 322–346.

Kurosawa, K. and Ogata, W. 2004. Oblivious keyword search. *J. Complex. 20,* 2–3, 356–371.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. 2004. RCV1: A new benchmark collection for text categorization research. *J. Mach. Learn. Resea. 5*, 361–397.

Long, D. 2002. Improving Information Retrieval System Security via an Optimal Maximal Coding Scheme. In *Proceedings of the 1st EurAsian Conference on Information and Communication Technology (EurAsia-ICT'02)*.

Machanavajjhala, A., Gehrke, J., Kifer, D., and Venkitasubramaniam, M. 2006. L-Diversity: Privacy beyond K-anonymity. In *Proceedings of the IEEE International Conference on Data Engineering*. 24–35.

Menon, S. and Sarkar, S. 2007. Minimizing information loss and preserving privacy. *Manag. Sci. 53,* 1, 101–116.

Meyerson, A. and Williams, R. 2004. On the complexity of optimal K-anonymity. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'04)*, 223–228.

Ostrovsky, R. and Skeith, W. E. 2007. Private searching on streaming data. *J. Cryptol. 20,* 4, 397–430.

Pang, H. and Mouratidis, K. 2008. Authenticating the query results of text search engines. In *Proceedings of the Very Large Data Bases*. 126–137.

PANG, H., TAN, K.-L., AND ZHOU, X. 2004. Steganographic schemes for file system and B-tree. *IEEE Trans. Knowl. Data Engin. 16,* 6, 701–713.

ROBERTSON, S. E. AND JONES, K. S. 1976. Relevance weighting of search terms. *J. Amer. Soc. Inform. Sci. 27,* 3, 129–146.

ROBINSON, J. T. 1981. The K-D-B-Tree: A search structure for large multidimensional dynamic indexes. In *Proceedings of the ACM International Conference on Management of Data*. 10–18.

ROSARIO, B. 2000. Latent semantic indexing: An overview. Techn. rep. INFOSYS 240 Spring Paper, University of California, Berkeley. http://www.sims.berkeley.edu/ rosario/projects/LSI.pdf.

SALTON, G. 1989. *Automatic Text Processing—The Transformation, Analysis, and Retrieval of Information by Computer*. Addison–Wesley.

SALTON, G. 1991. Developments in automatic text retrieval. *Sci. 253,* 5023, 974–979.

SAMARATI, P. 2001. Protecting respondents' Identities in microdata release. *IEEE Trans. Knowl. Data Engin. 13,* 6, 1010–1027.

SANDHU, R. S., COYNE, E. J., FEINSTEIN, H. L., AND YOUMAN, C. E. 1996. Role-based access control models. *IEEE Comput. 29,* 2, 38–47.

SHEN, X., TAN, B., AND ZHAI, C. 2007. Privacy protection in personalized search. *ACM SIGIR Forum 41,* 1, 4–17.

SONG, D. X., WAGNER, D., AND PERRIG, A. 2000. Practical techniques for searches on encrypted data. In *Proceedings of the IEEE Symposium on Security and Privacy*. 44–55.

SWEENEY, L. 2002. k-Anonymity: A model for protecting privacy. *Int. J. Uncertain., Fuzz. Knowl.-Based Syst. 10,* 5, 557–570.

VAN RIJSBERGEN, C. 1986. A non-classical logic for information retrieval. *Comput. J. 29,* 6, 481–485.

VAN RIJSBERGEN, C. J. 1979. *Information Retrieval*. Butterworth.

WANG, L., NOEL, S., AND JAJODIA, S. 2006. Minimum-cost network hardening using attack graphs. *Comput. Commun. 29,* 18, 3812–3824.

WANG, S., DING, X., DENG, R. H., AND BAO, F. 2007. Private information retrieval using trusted hardware. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*. 49–64.

XIAO, X. AND TAO, Y. 2006. Personalized privacy preservation. In *Proceedings of the ACM International Conference on Management of Data*. 229–240.

YAN, H., GROSKY, W. I., AND FOTOUHI, F. 2008. Augmenting the power of LSI in text retrieval: Singular value rescaling. *Data Knowl. Engin. 65,* 1, 108–125.

ZHONG, S., YANG, Z., AND WRIGHT, R. N. 2005. Privacy-enhancing k-anonymization of customer data. In *Proceedings of the ACM Symposium on Principles of Database Systems*. 139–147.

ZOBEL, J. AND MOFFAT, A. 1995. Adding compression to a full-text retrieval system. *Softw.—Practice Exper. 25,* 8, 891–903.

ZOBEL, J. AND MOFFAT, A. 2006. Inverted files for text search engine. *ACM Comput. Surv. 38,* 2, 6.

Received October 2008; revised January 2009, June 2009; accepted November 2009