PrivatePond: Outsourced Management of Web Corpuses

Daniel Fabbri
dfabbri@umich.edu
University of Michigan

Arnab Nandi
arnab@umich.edu
University of Michigan

Kristen LeFevre
klefevre@umich.edu
University of Michigan

H.V. Jagadish
jag@umich.edu
University of Michigan

ABSTRACT

With the rise of cloud computing, it is increasingly attractive for end-users (organizations and individuals) to outsource the management of their data to a small number of large-scale service providers. In this paper, we consider a user who wants to outsource storage and search for a corpus of web documents (e.g., an intranet). At the same time, the corpus may contain confidential documents that the organization does not want to reveal to the service provider.

While past work has considered the problems of secure keyword search and secure indexing, all of the proposed tools require significant modifications to existing search engines and infrastructure. In this paper, we propose a system called PrivatePond, which allows confidential outsourced web search using an unmodified search engine. The system is built around the central idea of a *secure indexable representation*, which is attached to each document in the corpus, and constructed with the goal of balancing confidentiality and searchability. In addition, a secure local proxy is used to provide transparency to the end-user.

While the idea of a secure indexable representation is very general, we propose a preliminary instantiation of this idea, which provides practical confidentiality. In addition, an experimental evaluation indicates that this indexable representation can provide high-quality search and ranking, similar to what is available using the unmodified corpus.

1. INTRODUCTION

In the era of cloud computing, it is increasingly common for individuals and organizations alike to outsource their data management to a small number of centrally-managed highly-scalable service providers. In this paper, we consider an end-user (organization or individual) who wants to outsource storage and search for a corpus of confidential web documents (i.e., a set of text or HTML pages connected to one another by hyperlinks).

This problem arises in a variety of settings. As a concrete example, consider an organization that would like to outsource management of its intranet to a third-party service provider. This is attractive for the organization because it saves the time, money, and human resources that are required to manage the infrastructure in-house. Instead, users within the organization can be authenticated internally, and their requests re-directed to the service provider. For small companies, this is also simpler and less-expensive than, for example, purchasing a Google search appliance [1]. In order for this setup to be practical, however, the clear challenge is security. Corporate intranets often contain sensitive and proprietary documents, information that should not be shared directly with an external service provider.

We propose a novel system called PrivatePond, which was designed with the goal of allowing an end-user to create, store, and search a corpus of web documents, using an untrusted service provider, and without compromising the confidentiality of the documents in the corpus. We deliberately sought out a pragmatic system design, identifying the following practical requirements:

1. **Existing infrastructure:** The system should be able to leverage existing web search and information retrieval infrastructure (e.g., indexing and ranking) without modification.
2. **Confidentiality:** The system should provide practical confidentiality guarantees. In particular, the service provider is not trusted to see the documents in the original corpus (in unencrypted form).
3. **Search quality:** The system should leverage the many years of research in web search and IR ranking algorithms in order to provide precision, recall, and ranking quality similar to standard (insecure) search.
4. **Minimal overhead:** The system should minimize client-side pre-processing and post-processing costs. In the pathological case, an end-user could simply encrypt his entire private corpus (e.g., as a BLOB), and store it with the service provider. However, each time the end-user wants to perform a search, he has to fetch the encrypted BLOB, decrypt it on the client side, and then perform the search locally. This approach is unsatisfactory because it requires significant client-side post-processing, and it fails to leverage the service provider's existing search infrastructure.
5. **Transparency:** The system should provide transparent and seamless interactions with the end-user.

Supported in part by NSF grant IIS-0438909, a gift from Yahoo!, and a seed grant from the University of Michigan.

Twelfth International Workshop on the Web and Databases (WebDB 2009), June 28, 2009, Providence, Rhode Island, USA.

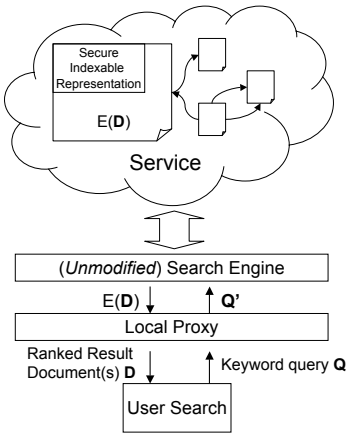


Figure 1: PrivatePond Architecture

1.1 Related Work

A significant body of related work has focused on two key problems that are closely related to ours: securely performing boolean keyword search on (one or more) encrypted documents and secure inverted indexes for document retrieval and ranking.

Song et al. [21] presented the first cryptographic scheme for securely performing keyword search on one or more outsourced documents. The protocol is based on stream ciphers, and provides very rigorous security guarantees (specifically, the untrusted search provider learns nothing about the documents except the search results). However, to search a single document of length n requires $O(n)$ operations, which means that the techniques do not practically scale to large corpuses. The work also only considers boolean keyword search, rather than web and IR-style search and ranking (e.g., PageRank or TF-IDF). Boneh et al. developed cryptographic tools for secure keyword search in public key encryption [5].

Recently, there has also been significant interest in developing techniques to secure inverted indexes (for searching a corpus of private documents). However, each of the proposed systems and indexing structures require significant modifications be made to the existing indexing and search infrastructure.

Among the first secure indexing tools, the μ-Serve system was developed at IBM to index access-controlled documents distributed over an intranet [4]; the system provides a centralized index based on Bloom filters, which responds to keyword queries by returning a superset list of documents containing the key terms. Goh [9] also developed a secure index using Bloom filters, making somewhat stronger cryptographic guarantees, and Chang and Mitzenmacher [7] also developed an encrypted keyword index. In addition to requiring new infrastructure, none of these proposals considered IR- or web-style result ranking.

More recently, the Zerber system [24, 25] was proposed, also with the goal of securely searching access-controlled documents within a corporate intranet. Like our work, Zerber considers a statistical attack based on document frequencies, and makes a probabilistic guarantee (r-confidentiality) [24]. Zerber also supports IR-style ranking based on term frequencies (TF) [25]. However, the mechanism used to enforce the statistical privacy requirement (merging posting lists) still requires modifications to the indexing infrastructure. Work by Swaminathan et al. [22] also provides secure ranking using TF, but the solution is based on precomputing each term frequency, and storing these in encrypted form as part of this index, thus requiring significant modifications to the index as well as search processing.

In addition to text and web data, recent research has also considered security problems that arise when outsourcing relational databases to untrusted service providers. Hacigümüş et al. describe a scheme for evaluating SQL queries on encrypted relational data, which partitions the query plan into pieces, some of which are evaluated by the service provider, and some by the client [11]. Agrawal et al. developed an order-preserving encryption scheme aimed at improving the efficiency of range queries in this setting [2]. Much work has also focused on guaranteeing query result integrity and completeness in this setting [8, 14, 18, 19, 23].

Finally, research has developed techniques for controlling access to published databases using cryptography [10, 15], but this work did not consider searching or querying the encrypted data.

1.2 Contributions

In this paper, we present the design, implementation, and evaluation of PrivatePond. The system is highlighted by the following important features:

- The system is built around a novel, pragmatic, and generic system architecture, described in Section 2, which does not require any modifications to existing search engine infrastructure (i.e., indexing and search mechanisms). A key concept in the proposed architecture is to associate with each document a *secure indexable representation*, which is intended to balance the goals of searchability and confidentiality. In addition, the architecture includes a secure *local proxy*, which hides changes in the search workflow from the end-user.
- We provide a practical instantiation of the secure indexable representation in Section 3, which we analyze both in terms of security and searchability.
- Finally, we describe the results of an extensive experimental evaluation in Section 4. In particular, the experiments indicate that by using PrivatePond (and the secure indexable representation) it is possible to achieve both practical confidentiality and high-quality search.

2. SYSTEM DESIGN

This section describes our basic architecture for outsourcing storage and search for a corpus of web documents. We consider the setting in which the service provider is trusted to maintain the integrity of outsourced data, and to reliably answer queries correctly. However, the service provider is not trusted to maintain the confidentiality of the corpus.

Consider an end-user with a corpus of web data $C = (\{D_1, ..., D_n\}, \{L\})$, where $D_1, ..., D_n$ are HTML documents and $\{L\}$ is a set of hyperlinks. When outsourcing corpus C, we create an alternative representation of C (denoted C') with the goal of protecting confidential information in C while simultaneously supporting search using the unmodified search architecture.

We create C' from C in the following manner. First, we would like to fully recover each document D, while hiding its content from the service provider. This is accomplished by encrypting the document to produce $E(D)$.[1] Of course,

[1] $E()$ can be any symmetric-key encryption algorithm, where

$E(D)$ is not searchable. For this reason, we also create an additional *indexable representation*, $I(D)$, for each document D in the corpus. $I(D)$ is attached or appended to $E(D)$ to allow the search engine to index and search the data, while limiting the information revealed about D. In general, $I(D)$ will be some encrypted form of the words in the document; clear text is unacceptable as confidential information would be released. Lastly, we produce a new set of links L', which is a subset of L, that contains all links between the documents in the corpus; external links are stored within the encrypted documents, but are not available for search. The resulting corpus $C' = (\{(E(D_1), I(D_1)), ..., (E(D_n), I(D_n))\}, \{L'\})$ is then given to the service provider, where it can be indexed and searched. We present a particular example of an indexable representation in Section 3.

Figure 1 shows the basic system architecture. The service provider's search engine remains unchanged, but a proxy is inserted between the end-user and the search engine. The proxy is placed within the end-user's trusted computing platform (e.g., as part of the browser or in a trusted web domain), stores the key for encryption and decryption and is responsible for modifying keyword queries so that the queries issued to the search engine are consistent with the indexable representations $I(D)$. When a query Q is issued, the query is first sent to the proxy, which modifies the query, and sends the new query Q' to the service provider. The service provider uses an unmodified search engine to retrieve a ranked list of documents. The proxy then decrypts the encrypted representation of the documents and presents the text to the user.

This architecture is general, and we have intentionally left the idea of an indexable representation open. Various indexable representations are possible, each with its own security guarantees and impact on search quality. In the next section, we describe a particular instantiation of this idea.

3. SAMPLE INDEXABLE REPRESENTATION

The architecture described in the previous section is very general. In this section, we describe a sample indexable representation $I(D)$. Of course, the goals are two-fold. The indexable representation should protect the confidentiality of the original document. (In particular, we want to prevent an attacker from learning the original value of any token in $I(D)$.) Simultaneously, we want to preserve the quality of the results for searches evaluated on C'.

3.1 Constructing a Confidential Indexable Representation

We considered various techniques to produce a secure and searchable representation. As a simple strawman, an indexable representation could be created by encrypting each *token* (i.e., a whitespace-delineated word). Specifically, for each document $D_i = [T_1, ..., T_N]$ in the corpus, where T is a token in the document, we could produce an indexable representation $I(D_i) = [E(T_1), ..., E(T_N)]$. This construction prevents an adversary (like a malicious service provider) from easily learning the original tokens since they are no longer presented in the clear. However, the indexable representation is still vulnerable to attacks that utilize background information about the language. In the remainder of this section, we describe such attacks, as well as some practical modifications that can be made to the indexable representation.

each end-user is in possession of private key k. If the end-user is concerned about revealing the length of each document, he can also include additional padding in each $E(D)$.

Attacks on a Single Indexable Representation Unfortunately, even in its encrypted form, an adversary may be able to learn the value of a token in the indexable representation $I(D_i)$ using a language model that contains information about the structure and frequency of tokens in documents.

We consider two specific attacks on a single indexable representation using the strawman. First, since the order of tokens is maintained, the adversary could use a natural language analysis (e.g. <noun> <verb> <noun>) to determine possible values of each encrypted token. Thus, we remove the order of tokens in the indexable representation.

Second, even with the order of tokens removed, the indexable representation is still vulnerable to attacks. In particular, since the frequency of the tokens in a document is maintained, and the adversary has knowledge about the relative frequency of tokens in the language, the adversary can sometimes deduce the values of some encrypted tokens. For example, if the adversary knows that the word *the* is the most common word in the language and the encrypted token t is the most frequent in the outsourced indexable representation, then the adversary can determine that token t has value *the*. For example, Kumar et al. show that a hashed data set is vulnerable to attacks that consider the frequency of tokens in a document [13]. Therefore, we only allow each token to occur once per indexable representation.

To prevent these attacks on a single indexable representation, we reduce the amount of information in $I(D_i)$. Specifically, given a document D_i in the corpus C, we produce an indexable representation $I(D_i) = \{E(T_j), E(T_k), ..., E(T_l)\}$ that is an unordered set of tokens (without duplicates). This construction prevents the previous two attacks since the tokens are not organized in any specific order and each unique value occurs at most once.

Attacks on a Corpus of Indexable Representations The indexable representation described above protects the confidentiality of the indexable representation for a single document. Unfortunately, the adversary may still leverage the aggregate information from all documents in the outsourced corpus C'. Specifically, the adversary can construct the frequency of tokens in the corpus by analyzing the *document frequency* of each token (i.e., the number of documents the token occurs in). Then, using the document frequency and the background information about the relative frequency of tokens in the language, the adversary can determine the value of encrypted token in the indexable representation; this is the same attack that can be performed on a single indexable representation if token frequencies are maintained.

In the worst case, we can take a pessimistic view, and suppose that the adversary knows the document frequency of each token in the corpus. In this case, a privacy breach could occur if any token has a unique document frequency in C'. More generally, we define the *bin width* (BW) as the number of tokens with the same document frequency in the outsourced corpus and require that each token have the same document frequency of at least *bin width* $- 1$ other tokens. To limit the adversary's ability to determine the value of an encrypted token, we modify the contents of the corpus.

One way to produce a secure outsourced corpus C' is to

increase the document frequency of tokens that already exist in the corpus C. Specifically, we sort the tokens by document frequency order. Then, we process the tokens in decreasing order. If token T_i does not have the required bin width, we add the next token with lesser document frequencies, T_{i-1}, randomly to other documents' indexable representation until token T_{i-1} has the same document frequency as token T_i. We repeat this process until the corpus is secured. For example, if the original corpus has three tokens $\{a, b, c\}$ with document frequencies $\{1, 2, 3\}$ and a *bin width* $= 3$, then we would add token a to two other documents and token b to one other document. This padding procedure is performed during a pre-processing step, and the user can specify the amount of binning that is required.

3.2 Search Quality Analysis

Unfortunately, securing the data in the indexable representation limits our ability to conduct conventional search and ranking. By eliminating the order of terms (viewing documents as a set-of-words), we have eliminated the possibility of conducting *proximity-based* searches. Similarly, because we eliminated duplicate terms from each indexable representation, we are no longer able to compute single-document term-frequencies (the TF component of the standard TF-IDF ranking function). In this case, each TF term is either 1 or 0.

Similarly, to reduce the likelihood of a language model-attack across multiple documents, we have chosen to introduce additional terms (padding). As the number of additional inserted terms increases, we expect that this will introduce additional false positives into even simple boolean keyword searches. We examine the impact of padding experimentally in Section 4.

Finally, in web search, ranking is not typically based entirely on document content, but often also on the link structure of the corpus. Link-based ranking algorithms (e.g., PageRank [6]) are unaffected by the set-of-words representation because we maintain the internal link structure within the corpus C'; however, the random padding of terms may introduce false positives if a term is added to a page with a high page rank.

4. EVALUATION

To evaluate our ideas, we use a sample of the Simple English Wikipedia (`http://simple.wikipedia.org`); a collection of 3550 documents with over 10,800 links between them. We refer to this as the *small corpus* (*Small C*). Additionally, we evaluate the entire Simple Wikipedia corpus; a collection of over 43,000 documents with over 800,000 links between them. We refer to this as the *large corpus* (*Large C*).

A query workload of approximately 10,000 keyword queries, ranging from one to three words in length, was generated by randomly sampling the corpuses to evaluate search quality.

The primary goal of our experimental study was to measure the impact of the secure indexable representation on search quality. While performance may be a secondary concern, for our corpus, the time necessary to generate indexable representations was only a few minutes on a dual core CPU system with 2GB of RAM, running Red Hat Linux. We did not see any significant difference in query performance between the original and outsourced versions of our corpus for all bin widths up to 1000.

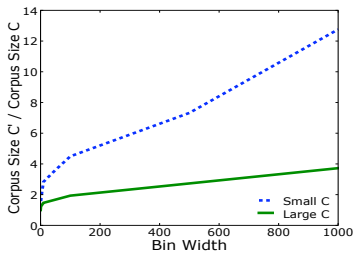


Figure 2: Corpus Size vs Bin Width

4.1 Effects of Bin Width on Corpus Size

As bin width increases, more tokens are added to documents in the corpus. In the worst case each document will contain every word in the corpus (i.e. the vocabulary), and hence the padded corpus size is equal to $|vocabulary| \times |corpus|$. We study the increase in the size of the outsourced corpus C' (in tokens) to the original corpus C.

As we can see in Figure 2, the outsourced corpus C' is larger than the original corpus when there is no padding because we add a secure indexable representations to each document. For larger bin widths and the small corpus, the outsourced corpus size increases to nearly three times the size of the original corpus for *bin width* $= 10$ and five times larger for *bin width* $= 100$. In contrast the larger corpus grows at a slower rate as the bin width increases because less padding is needed to meet the desired bin width.

4.2 Search Quality

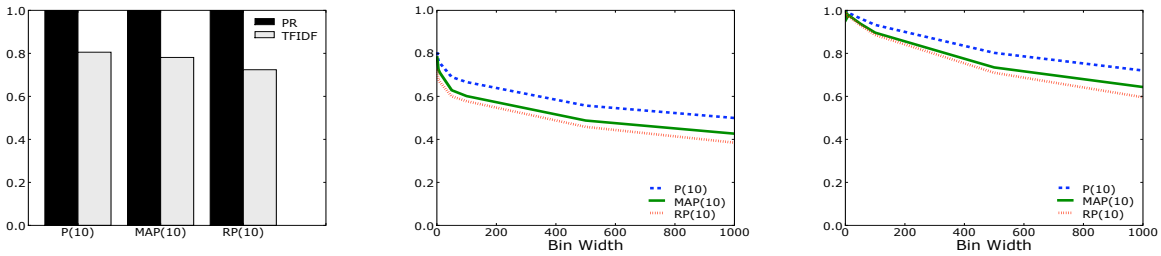
We examine the costs of secure search on confidential information using an unmodified search engine. There is a tradeoff between securely searching a corpus of documents and providing the highest search quality. Clearly, the secure indexable representation impacts search quality since token frequencies are modified. Next, we evaluate the effects of the secure indexable representation and the bin width security requirement on search quality.

4.2.1 Ranking Models

To evaluate the search quality, we test our system with two families of ranking models.

- **Content-Based Ranking:** First, we consider an IR ranking model where the score of a document depends on the contents of the document. Examples include term frequency (TF) and term frequency-inverse document frequency (TFIDF). We study MySQL's FULLTEXT search implementation [20] that considers TF of the queried term and IDF of all the terms in the documents. For the scope of this paper, we will call this method TFIDF (even though it is not just simply TF $\times$ IDF).
- **Global Ranking:** Second, we consider a global ranking model where the score is independent of the contents. We use PageRank [17] as our global ranking model for our hyperlinked collection of documents, using $d = 0.85$.
- **Combination of Ranking Models:** We expect a practical and state-of-the-art search system to include a combination of the two ranking models.

Both models have their own sensitivities to the indexable representation, namely switching to the set-of-words representation and the padding of additional words. As discussed in 3.2, content-based ranking models (such as our



(a) Search Quality, BW = 1 (b) TFIDF Search Quality vs BW (c) PageRank Search Quality vs BW

Figure 3: Search Quality, Small Corpus

TFIDF variant) consider frequencies of words and the length of the document as features, both of which are modified when converting documents to sets-of-words. Additionally, both models are affected by the padding of additional tokens; padded tokens can create false positives for search: a search for 'x y' brings up a document with original content 'x' because 'y' was added to it.

We assume that each search result contains all terms from the query (boolean conjunctive search), and that results are presented in ranked order, starting with results with the highest ranking score.

4.2.2 Metrics for Search Quality

When we modify our corpus for the purpose of generating a secure indexable representation, we try to ensure three key factors: (1) minimize loss of search results, (2) minimize loss of *important* (top ranking) search results, and (3) minimize loss of ranking information. Next, we describe our procedure for evaluating the search quality of an indexable representation.

Given a query, we first determine the list of documents returned when the original (unencrypted) corpus is searched; we will call this the *gold* list. Next, we use the same unmodified search engine to search the corpus of indexable representations to produce the *pond* list, and then measure the difference between the gold list and pond list. For example, if our original search engine lists documents $[p, q, r]$ as the top-3 results in ranked order, the indexable representations may generate $[q, s, p]$ as the top-3. Clearly, r is now missing, while q and p have changed ranks. We use three metrics to measure the change in search quality:

- **Precision at N (P):** First, we measure the number of documents returned by the gold list that are contained in the pond list. The precision at N is defined as:

$$P(N) = \frac{|gold_{[1,N]} \cap pond_{[1,N]}|}{N}$$

  where the subscripts specify the set of documents we consider from each list. If every gold document is returned, the precision is 1.0; if every gold document is missing, the precision is 0.0. From the example in the previous paragraph, $P(3) = \frac{2}{3}$, which means intuitively that the pond list contained two of the top-3 results from the gold list.

- **Mean Average Precision (MAP):** Second, we use the metric mentioned in [16] to examine the precision at various depth for the pond list. The mean average precision is calculated with:

$$MAP(N) = \frac{\sum_{r=1}^{N} (\frac{|gold_{[1,N]} \cap pond_{[1,r]}|}{r} \times rel(r))}{N}$$

  where $rel(r)$ is 1 if the r^{th} document in the pond list exists in the gold list, otherwise 0. Like precision at N, MAP ranges from 0.0 to 1.0, but punishes the precision score if documents that are not contained in the top-N of the gold list are ranked higher than documents in the gold list. For the example above, MAP(3) $= \frac{\frac{1}{1}+0+\frac{2}{3}}{3} = \frac{5}{9}$, which is less than P(3) since document s was ranked higher than p.

- **Rank Perturbation (RP):** Third, we consider the change in rank between the gold list and the pond list. We define the rank perturbation as:

$$RP(N) = 1 - \frac{\sum_{i=gold[1]}^{gold[N]} |g_i - p_i|}{N^2}$$

  where g_i is the rank of document i in the gold list and p_i is the rank of document i in the pond list; if a document is contained in the gold list but is missing from the pond list, the score for the specific document is N. For the example above, $RP(3) = 1 - \frac{2+1+3}{9} = \frac{1}{3}$, which is worse than P(3) and MAP(3) since rank perturbation takes into account the change in rank of p and q and that r is missing. We considered using Spearman's distance or Kendall's tau, however, these metrics are undefined when the gold and pond lists contain different documents.

4.2.3 Effects of the Secure Indexable Representation

First, we evaluate the search quality for the set-of-words indexable representation with our query workload. We present the average score for $N = 10$ for all metrics. The representation itself reduces search quality for algorithms that rely on the frequency of terms in a document, such as TFIDF. For models that do not use the term frequency, such as PageRank, there is no effect due to the representation.

In Figure 3(a), we analyze the change in search quality for the TFIDF and PageRank models with the small corpus. We observe that moving to a set-of-words representation incurs a drop in quality even with the minimum possible bin width of 1 for the TFIDF model; P(10) tells us that on average, 2 of of the gold's top-10 search results will be beyond the pond's top-10, while MAP and RP also show a further decline in search quality due to differences in ranking. In contrast, the PageRank model is unaffected as expected; PageRank returns every document in the gold list.

4.2.4 Effects of Bin Width

We evaluate the change in search quality for TFIDF and PageRank as the bin width increases. Figure 3(b) and Figure 3(c) show there is a very slow but gradual loss of quality as we increase the bin width for both ranking models because false positives are introduced due to padding documents with a high PageRank or low document frequency.

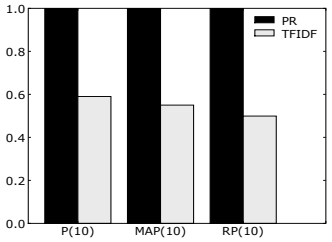


Figure 4: Search Quality, Large C, BW = 1

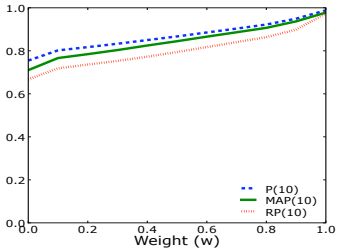


Figure 5: TFIDF + PageRank, Small C, BW = 10

4.2.5 *Effects of a Larger Corpus*

We analyze how search quality changes given a larger data set. Figure 4 shows that the set-of-words representation for the large corpus has worse TFIDF search quality than the smaller corpus (Figure 3(a)); P(10) indicates that six of the top-10 documents from the gold list are returned on average. The poorer ranking is a result of having more documents with the same tokens in their indexable representations. Like the smaller corpus, the PageRank results are unaffected by the set-of-words representation. For bin widths up to 500, the search quality does not decrease; for larger bin widths, the search quality decreases in a similar manner to the smaller corpus (Figure 3(b), 3(c)). We do not include these plots due to space constraints.

4.2.6 *Effects of Combining TFIDF and PageRank*

In addition to studying the search quality of the two ranking models independently, we present the quality of search results with *bin width* = 10 for a combination of the two models using a weighted sum: $(w) \cdot (PageRank) + (1-w) \cdot (TFIDF)$. We consider various values of w and evaluate the change in search quality on the small corpus.

As expected, we can see in Figure 5 that there is a clear tradeoff between global ranking such as PageRank and content-based ranking such as TFIDF. A w of 0.75 misses less than one document per search on average and can be considered an acceptable parameter for most search applications.

5. CONCLUSION & FUTURE WORK

We present the PrivatePond architecture, a practical implementation of secure search on confidential information using an unmodified search engine. We consider the tradeoffs between data security and search quality when outsourcing a corpus of documents to a service provider. We examine a sample indexable representation that is resistant to adversarial attacks, but allows for search quality to be comparable to that of the original corpus. Alternative indexable representations are possible in the PrivatePond architecture with varying search quality and confidentiality characteristics. In the future, we will consider the network structure of the corpus as a source of uniquely identifiable information [3, 12].

6. REFERENCES

[1] Google search solutions for business. http://www.google.com/enterprise/search/gsa.html.
[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *SIGMOD*, 2004.
[3] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, 2007.
[4] M. Bawa, R. Bayardo, and R. Agrawal. Privacy-preserving indexing of documents on the network. In *VLDB*, 2003.
[5] D. Boneh, G. Crescenzo, R. Ostravsky, and G. Persiano. Public-key encryption with keyword search. In *Eurocrypt*, 2004.
[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW*, 1998.
[7] Y.-C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security*, pages 442–455. 2005.
[8] P. Devanbu, M. Gertz, C. Martel, and S. Stubblebine. Authentic third-party data publication. In *IFIP 11.3 Workshop on Database Security*, 2000.
[9] E. Goh. Secure indexes. Cryptology ePrint Archive, 2004.
[10] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS*, 2006.
[11] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *SIGMOD*, 2002.
[12] M. Hay, G. Miklau, D. Jensen, D. Towsely, and P. Weis. Resisting structural re-identification in anonymized social networks. In *VLDB*, 2008.
[13] R. Kumar, J. Novak, B. Pang, and A. Tomkins. On anonymizing query logs via token-based hashing. In *WWW*, 2007.
[14] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin. Dynamic authenticated index structures for outsourced databases. In *ACM SIGMOD*, 2006.
[15] G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *VLDB*, 2003.
[16] A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Trans. Inf. Syst.*, 27(1):1–27, 2008.
[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1998.
[18] H. Pang, A. Jain, K. Ramamritham, and K. Tan. Verifying completeness of relational query results in data publishing. In *ACM SIGMOD*, 2005.
[19] H. Pang and K. Tan. Authenticating query results in edge computing. In *ICDE*, 2004.
[20] A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document length normalization. *Information Processing and Management*, 32(5):619–633, 1996.
[21] D. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE Security and Privacy Symposium*, 2000.
[22] A. Swaminathan, Y. Mao, G. Su, H. Gou, A. Varna, S. He, M. Wu, and D. Oard. Confidentiality-preserving rank-ordered search. In *StorageSS*, 2007.
[23] M. Xie, H. Wang, J. Yin, and X. Meng. Integrity auditing of outsourced data. In *VLDB*, 2007.
[24] S. Zerr, E. Demidova, D. Olmedilla, W. Nejdl, M. Winslett, and S. Mitra. Zerber: r-confidential indexing for distributed documents. In *EDBT*, 2008.
[25] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski. Zerber+r: Top-k retrieval from a confidential index. In *EDBT*, 2009.