Privacy-Preserving Indexing of Documents on the Network

Mayank Bawa   Roberto J. Bayardo Jr.   Rakesh Agrawal

Stanford University
Stanford, CA 94305
bawa@db.stanford.edu

IBM Almaden Research Center
San Jose, CA 95120
{bayardo,ragrawal}@almaden.ibm.com

Abstract

We address the problem of providing privacy-preserving search over distributed access-controlled content. Indexed documents can be easily reconstructed from conventional (inverted) indexes used in search. The need to avoid breaches of access-control through the index requires the index hosting site to be fully secured and trusted by by all participating content providers. This level of trust is impractical in the increasingly common case where multiple competing organizations or individuals wish to *selectively* share content. We propose a solution that eliminates the need of such a trusted authority. The solution builds a centralized *privacy-preserving* index in conjunction with a distributed *access-control enforcing* search protocol. The new index provides strong and quantifiable privacy guarantees that hold *even* if the entire index is made public. Experiments on a real-life dataset validate performance of the scheme. The appeal of our solution is two-fold: (a) Content providers maintain complete control in defining access groups and ensuring its compliance, and (b) System implementors retain tunable knobs to balance privacy and efficiency concerns for their particular domains.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003

1 Introduction

While private and semi-private information on the network has grown rapidly in recent years, mechanisms for searching this information have failed to keep pace. A user faced with the problem of locating an access-controlled document must typically identify and individually search each relevant repository, assuming of course the user knows and remembers which repositories are relevant!

The lack of tools for searching access-controlled content on the network stems from the considerable difficulty in creating a search-engine that indexes the content while respecting the security and privacy requirements of the content providers. Contemporary search engines [4, 16, 22] build inverted indexes that map a keyword to its precise locations in an indexed document. The indexed document can thus be easily reconstructed from the index. Conferred with knowledge of *every* searchable document, the trust required of a search engine over access-controlled content grows rapidly with each participating provider. This enormous trust requirement, coupled with the potential for a complete breach of access control by way of malicious index disclosure, render such an approach impractical.

In this paper we address the problem of providing an efficient search mechanism that respects privacy concerns of the participating content providers. Our solution is to build a centralized index of content that works in conjunction with an *access control enforcing* search protocol across networked providers. The centralized index itself provides strong and quantifiable privacy guarantees that hold *even* if the entire index is made public. The degree of privacy provided by the index can to be tuned to fit the needs of the providers, and overhead incurred by the search protocol is proportional to the degree of privacy provided.

We envision applications of this technology in various sectors, where multiple organizations are actively competing as well as collaborating with constantly evolving alliances. Another application domain is file-sharing through personal webservers (e.g., YouServ [18]). For example, our scheme could be used by individuals to share copyrighted songs electronically with others who can authenticate they already own the song. The providers can keep track of the proofs supplied to allow audit of such exchanges.

Our method of providing efficient search over access-controlled content preserves the important appeal of private information sharing — each provider has complete control over the information it shares: how much is shared, when it is shared, and with whom it is shared.

Layout: Section 2 defines the problem, privacy goals and an adversary model. Section 3 presents privacy attacks on conventional search solutions. Section 4 defines a privacy-preserving index structure. Section 5 provides a mechanism for constructing such an index. Section 6 evaluates performance of the index on a real-life dataset. Section 7 explores related work. Section 8 concludes the paper.

2 Preliminaries

In this section, we define the problem of searching distributed access-controlled content and the assumptions our solution makes on the supporting infrastructure. We also present the privacy goals that are the focus of this paper, followed by a privacy spectrum for characterizing the degree with which any solution achieves them.

2.1 Problem Statement

The input to the problem of searching distributed access-controlled content is a set of *content providers* $p_1, p_2, \ldots, p_n$, and a *searcher* s who issues a *query* q. Each provider is said to *share* a set of documents with access-control determined by the authenticated identity of the searcher s and an *access policy*. The desired output is the set containing documents d such that (1) d is shared by some provider p_i for $1 \leq i \leq n$, (2) d matches the query q and (3) d is accessible to s as dictated by p_i's access policy.

2.2 Assumptions on the Infrastructure

Most of the details of the query language, access policy language, and authentication mechanism are immaterial to our approach. We require only the following properties of each component:

A *Query language:* The query language must support conjunctive keyword queries. Additional constructs (e.g., phrase search, negated terms) can be supported as well, so long as they only further constrain the result set.

B *Authentication mechanism:* The authentication scheme should allow users to authenticate themselves to each content provider independently, preferably without requiring explicit registration with each provider. For example, client authentication through third-party signed security certificates (e.g., SSL/TLS [12, 7]) would be satisfactory.

C *Access policy language:* The only requirement of the access policy language is that content providers are themselves able to apply and enforce their access policies given the authenticated identity of the searcher. This allows, for example, each content provider to individually select a policy language that best fits its requirements.

2.3 Privacy Adversaries

Just as important as ensuring correct output for a query q is the requirement of preventing an *adversary* from learning what one or more providers may be sharing without obtaining proper access rights. We will characterize solutions to the problem in terms of their susceptibility to privacy breaches by the types of adversaries described here.

A *passive adversary* is an eavesdropper who merely observes and records messages (queries, responses, indexes) sent in the system. Such an adversary may have either a *global* (ability to observe all messages in the system) or a *local* (ability to observe messages sent to/from a particular content provider) view of the system. An *active adversary* is an entity which acts with deliberate intent in accordance with the system protocol to gather information. In our model, such an adversary could inspect index structures, issue various queries, or even participate in the index construction process to facilitate such breaches. Adversaries may also *collude* with each other to breach privacy.

Adversaries may also be categorized according to roles they can assume. For example, most users (and hence adversaries) will be limited to performing the role of a searcher since content providers are in practice likely to be a smaller and more controlled population. The information and operations accessible through each role (searcher, provider, indexer) can be used to facilitate different types of breaches.

2.4 Privacy Goal

We focus on attaining the following privacy goal with respect to a document d made searchable by some content provider p:

Content Privacy An adversary A should not be allowed to deduce that p is sharing some document d containing keywords q unless A has been granted access to d by p.

Other privacy goals related to distributed search but not addressed in this paper include *query privacy* and *provider anonymity*. Query privacy involves preventing an adversary from determining which searcher issued what particular queries. Provider anonymity involves preventing an adversary from determining which provider published what particular documents.

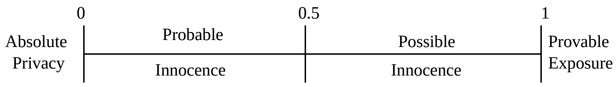

Figure 1: Degrees of privacy on a probabilistic scale of 0 (*Absolute Privacy*) to 1 (*Provable Exposure*)

2.5 Degrees of Privacy

To formally analyze a privacy-preserving scheme, we need to characterize the degree with which Content Privacy is attained against an adversary that does not have access to a document d being shared by provider p. To this end, we adapt the privacy spectrum used by Reiter and Rubin in their analysis of Crowds [23] as shown in Figure 1 and discussed below:

A *Provable Exposure:* The adversary can provide irrefutable evidence that p is sharing d.

B *Possible Innocence:* The claim of adversary about p sharing d can be false with a non-trivial probability (e.g., with probability in $(0.5, 1)$).

C *Probable Innocence:* The claim of adversary about p sharing d is more likely to be false than true (e.g., with probability in $(0, 0.5]$).

D *Absolute Privacy:* The adversary cannot determine if p is sharing d or not.

E *Beyond Suspicion:* The adversary cannot determine if p is more likely to be sharing document d than any other provider.

We can replace d in the above discussion by any set of keywords q, in which case our aim is to prevent the adversary from determining whether p is sharing a document that contains keywords in q.

3 Analysis of Conventional Solutions

In this section, we consider search solutions adopted by conventional systems and how they might be adapted to support search over access-controlled content. Such adaptations fail to address our privacy and efficiency goals, but their analysis provides insight into designing an improved search mechanism.

3.1 Centralized Indexing

The most common scheme for supporting efficient search over distributed content is to build a centralized inverted index. The index maps each term to a set of documents that contain the term. The index is queried by the searcher to obtain a list of matching documents. This is the scheme of choice of web search engines [4], mediators [16], and the now defunct Napster [22] network. The scheme can be extended to support access-controlled search by propagating access policies along with content to the indexing host. The index host must apply these policies for each searcher to filter search results appropriately. Since only the indexing host needs to be contacted to completely execute a search, searches are highly efficient.

Privacy Breaches: A centralized index will Provably Expose content providers to anyone who has access to the index structure. In cases where the index host is completely trusted by all content providers, this violation of access control may be tolerable. Unfortunately, finding such a trusted host is immensely difficult. Worse, compromise of the index host by hackers could lead to a complete and devastating privacy loss should the index be revealed publicly.

3.2 Query Broadcasting

At the other end of the efficiency spectrum lie broadcast-based schemes that send the query to *all* participating content providers. Such schemes include the Gnutella [14] network, where providers locally evaluate each query and directly provide any matching documents to the searcher. We can think of an augmented Gnutella-based search protocol that implements access control. In such a protocol, the query will be broadcast along with the identity and IP address of the query originator. Providers could securely deliver search results back to the authenticated searcher over an encrypted connection [12] to avoid interception. Since content shared by a provider p resides at p alone, providers are assured Absolute Privacy and the goal of Content Privacy is naturally preserved.

Performance Limitations: While the above adaptation to query broadcasting has excellent privacy characteristics, it suffers from poor scalability and severe performance penalties [24]. The protocols hence adopt heuristics (e.g., time-to-live fields) that limit search horizons and compromise search completeness.

3.3 Distributed Indexing

The performance limitations of query broadcasting have led to work on distributed indexing methods that support efficient search without the need for a single centralized index provider. KaZaa [19], for example, is a P2P network that leverages "super-peers" (machines with above-average bandwidth and processing power) by having them host sub-indexes of content shared by several less capable machines. The pSearch [28] system distributes a search index using a distributed hash table [21, 26]. In both these systems, the distributed index is used to identify a set of documents (or machines that host the documents) matching the searcher's query. These machines are then contacted directly by the searcher to retrieve the matching documents. Access control can be supported by simply having the providers enforce their access policies before providing the documents.

Privacy Breaches: Much as in the case of a centralized index, any node with access to a portion of the distributed index can Provably Expose any of the providers indexed by that portion. Worse, indexes are hosted by untrusted machines over whom the providers themselves have no control. An active adversary that does not host a portion of the index can search the distributed index to inflict privacy breaches. For example, the adversary can determine the precise list of providers sharing a document with a particular keyword by issuing a search on that keyword — a breach of Content Privacy with Provable Exposure. Content Privacy can also be breached by mounting *phrase attacks*. Such attacks take advantage of the observation that most documents have characteristic sets of words that are unique to them [6]. To identify a provider sharing some document, the adversary need only compose a query consisting of such terms for the document. The resulting list of sites are then known to share the document but with Possible Innocence. By choosing an appropriate set of terms, the adversary can achieve a near Provable Exposure.

3.4 Centralized Fuzzy Indexing

Some search applications do not maintain precise inverted index lists, but instead maintain structures that allow mapping of a query to a "fuzzy" set of providers that *may* contain matching documents. For example, YouSearch [2] builds a centralized Bloom filter [3] index. The Bloom filter index can be probed by a searcher to identify a list of all providers that contain documents matching the query. The list however is not necessarily precise, since bloom filters may produce *false positives* due to hash collisions. Given such a list, the searcher must contact each provider to accumulate results. These schemes can be extended to support access-controlled search by having the providers enforce their access policies at the point a searcher requests matching documents.

Privacy Breaches: Bloom filter indexes do offer limited privacy characteristics by virtue of potential false positives in the list of providers. Each provider in the list is thus Possibly Innocent of sharing a document matching the query. However, this privacy is spurious. An active adversary can perform a *dictionary-based attack* on the Bloom filter index to identify the term distribution of any indexed provider. Dictionary-based attacks take advantage of the fact that sentences in natural language (e.g., English) use words from a restricted vocabulary that are easily compiled (e.g., in a Oxford/Webster dictionary). Thus, the adversary can compute a hash for each word in the vocabulary. A provider in the Bloom filter entry for such a hash is, with some probability, sharing a document with the corresponding word. In addition, the scheme remains prone to phrase attacks.

4 A Privacy-Preserving Index (PPI)

Any search mechanism that relies on a conventional search index allows a provider to be Provably Exposed because of the precise information the index itself conveys. Efficient privacy-preserving search therefore requires an index structure that prevents Content Privacy breaches even in the event that the index is made public. In this section, we define such an index structure and analyze its privacy characteristics. We show that any index satisfying our definition leaves providers with at least Probable Innocence in response to active adversary attacks on the index structure. Section 5 presents a randomized algorithm for constructing such an index.

4.1 Search Methodology

While a conventional inverted list maps queries to lists of matching documents, an index that preserves privacy maps queries to lists of matching *providers*. Given the list of providers that may satisfy a query, it is then up to the searcher to directly query such providers and request matching documents. The providers, on receiving a query and authenticating the searcher, return a list of documents filtered according to the searcher's access rights.

By implementing search in this manner, we have moved the point of control from the index-hosting site to the providers. Providers can now manage and enforce access policies themselves without relying on any central host. While there is an efficiency penalty associated with the need to individually contact the providers, experimental results over publicly shared content [2] indicate the performance of such an approach can be quite reasonable in practice, even when there are many (> 1500) providers. [1]

4.2 Definition

A Privacy-Preserving Index (PPI) is a mapping function built on the set of documents D being shared by the set of providers $p_1, p_2, \ldots, p_n$. It accepts a query q and returns a subset of providers M that may contain matching documents. In order for the function to be considered *privacy preserving*, the set M for any query q must satisfy one of the following conditions:

A M is the *null set* only if there is no document in D that matches q.

B M is a *subset of providers* consisting of all providers that share a document matching q ("true positives") and an equal or greater number of providers that do not share a matching document ("false positives").

C M is the set of *all providers*.

[1] Organizations that share many documents could participate using multiple *virtual* providers where each virtual provider is responsible for handling search requests for a specific subrepository.

The PPI must behave like a conventional index: over time the index must return identical results for identical queries unless indexed content itself has changed. In addition, for any query q' whose results are a subset of another query q, the result set returned for q' must be a subset of that returned for q. These behavioral requirements prevent attacks that attempt privacy breaches by filtering out of false positives.

The PPI must be implemented with care: a naive implementation could easily yield more information than is provided by the PPI definition. For example, the indexing host might aggregate all shared content locally and preprocess it to materialize an index with true positives alone; the false positives as required by the definition being inserted into results at query time. Notice that in this case the materialized index itself does not correspond to PPI definitions. A public disclosure of the materialized index would result in Provable Exposure of content providers. Instead, we require that a materialized index should not yield any more information than that obtained from executing an exhaustive list of queries against the PPI.

4.3 Correctness

The set M returned by PPI for a query q never excludes any true positives for q. In other words, the result set for a query will contain all providers that have at least one matching document. The searcher contacts each provider to accumulate the results, who will release a document if and only if the searcher is allowed to access it. Thus, searching with a PPI leads to correct output.

4.4 Privacy Characteristics

Recall that the adversary who inspects the index has no advantage over the adversary issuing queries at will other than the time required for exploring the space of all queries. We can therefore restrict our analysis to the latter case.

Results for any query the adversary issues can correspond to one of Cases [A], [B] or [C] as defined above. If the result corresponds to Case [A], the adversary learns that no provider offers any document containing the specific term. All providers are Beyond Suspicion in that none is known to be more likely than the others to share such documents.

If the result corresponds to Case [B], at least half of the set of identified providers are false positives. Thus, all true positives within the set have Probable Innocence with respect to actually sharing a matching document. All providers outside the identified set are Beyond Suspicion.

If the result corresponds to Case [C], the adversary is unable to discriminate between providers. In effect, the index has degenerated into a broadcast-based mechanism, where all providers are Beyond Suspicion of sharing matching documents.

To relate privacy characteristics attained by the index with our goal of Content Privacy, we claim that by ensuring providers are always at least Probably Innocent for any inspection of the index by an active adversary, the adversary cannot bring about a strong privacy breach by exploiting a PPI alone. Note also that collusion between adversarial searchers offers no additional opportunities for privacy breaches.

4.5 Efficiency

Define *selectivity* σ of a query q to be the fraction of providers that share a document matching q. Observe that Case [B] causes the PPI to be at least $2\times$ less selective than an index which precisely maps queries to providers. Also observe that an optimally-efficient PPI must use Case [C] minimally: only for queries that have a selectivity $\sigma > 0.5$ that precludes them from Case [A] (trivially) and Case [B] (absence of an equal number of false positives). Hence, the PPI need not be more than $2\times$ less selective than a precise inverted index. Note however that there is an inherent trade-off between efficiency and the degree of Probable Innocence offered by a PPI. A PPI with optimal efficiency can never offer more than 50% false positives for queries resulting in Case [B] output.

An optimally efficient PPI yields the absolute minimum level of privacy (an adversary's claim is false with probability 0.5) required to satisfy the definition of Probable Innocence. Such a low degree of Probable Innocence may be inadequate for highly sensitive domains. Implementations should thus offer a means by which the level of false positives can be increased (at the expense of efficiency) to achieve a desired privacy level for the application domain. We discuss such an implementation scheme next.

5 Constructing a PPI

A procedure for constructing a PPI must address not only the correctness of the resulting structure, but also the potential for privacy breaches during the construction process. Ensuring privacy in the presence of adversarial participants is non-trivial since the index construction process involves pooling together information about content shared by each provider.

We now present a procedure to construct an index that is expected to be privacy-preserving for any particular query. The procedure partitions providers into "privacy groups" of size c. Within a group, providers are arranged in a ring. The providers execute a randomized algorithm which has only a small probability of error. We quantify this probability of error and show that by tuning a parameter, the error can be made small enough to be irrelevant in practice. We show that the construction process ensures that providers are resilient to breaches beyond Probable Innocence.

There are two exceptions where a provider may suffer a breach larger than Probable Innocence from adversaries *within* its privacy group. Providers who immediately precede an active adversary will be assured of only Possible Innocence with respect to sharing documents with a particular term. Specifically, an adversary neighbor can determine whether its predecessor along the ring is sharing a specific term with *at best* 0.71 probability.

The second exception is for a provider when both its neighbors along the ring collude against it. In such a case, the provider can be Provably Exposed as sharing documents containing particular terms. We will argue that such a breach can be minimized by having providers choose their two neighbors on the ring based on previously established real-world trust relationships.

5.1 Content Vectors

Our algorithm requires that each provider p_s summarize terms within its shared content through a bit vector V_s called its *content vector*. For example, a content vector might be a bloom filter [3] of system-specified length L which is formed as follows. Each provider p_s initializes its V_s by setting each bit to 0. Next, for each term t appearing in its shared content, the provider uses a system-specified hash function H with range $1, 2, \ldots, L$ to set position $H(t)$ in V_s to 1.

The content vectors thus formed are summaries of shared content at a provider. For example, a bloom filter V_s can be used to deduce if a provider p_s is sharing a document with term t as follows. We can hash t to a bit position in the bloom filter from provider p_s. If the bit is 0, then it is guaranteed that p_s shares no documents containing t. If the bit is 1, then the term might or might not occur at p_s since multiple terms might hash to the same value thus setting the same bit in V_s. The probability that such conflicts occur can be reduced by increasing the length L and/or using multiple hash functions.

5.2 Privacy Groups

The construction process starts by partitioning the space of providers into disjoint *privacy groups* of size $c > 2$ each. As we show later, the size of a privacy group is proportional to the degree of privacy enjoyed by each participant. Assume for now the partitioning scheme assigns members to groups at random.

For each privacy group G, the providers are arranged in a ring $p_1, p_2, \ldots, p_c$ (see Figure 2). We use the terms *successor* and *predecessor* of a provider in the usual way with respect to this ordering, with the additional requirement of p_1 being defined as the successor of p_c (and p_c the predecessor of p_1).

5.3 Group Content Vectors

Define the *group content vector* of a group G as the vector V_G resulting from performing a logical OR of the set of all content vectors from each provider in G. The next part of the construction is a randomized algorithm for generating the group content vector.

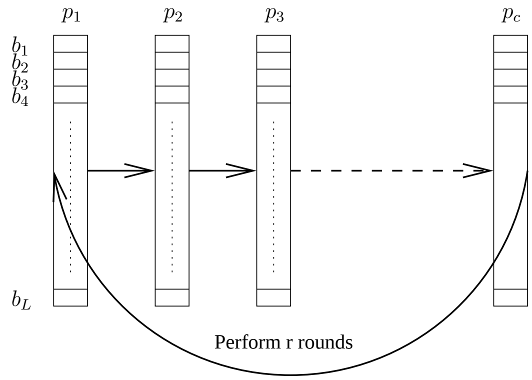

Figure 2: Index construction proceeds over r rounds.

IndexConstruction(r, V_s, V'_G)
$P_{ex} := 1/2^r$
$P_{in} := 1 - P_{ex}$
for $(i := 1; i < L; i := i + 1)$
do
if $(V_s[i] = 1$ *and* $V'_G[i] = 0)$
then Set $V'_G[i] := 1$ with prob. P_{in}
if $(V_s[i] = 0$ *and* $V'_G[i] = 1)$
then Set $V'_G[i] := 0$ with prob. P_{ex}
Send V'_G to *Successor*(s)

Figure 3: Processing of V'_G at p_s in step s of round r

The construction involves performing r rounds in which a vector V'_G is passed from provider to provider along the ring. Each provider, upon receiving the vector, performs the bit-flipping operations outlined in Figure 3 before passing the vector on to its successor. After r trips around the ring, the vector is sent to a designated index host.

The vector V'_G is initialized by p_1 to a vector of length L with each bit independently set to 0 or 1 with probability 1/2. Each round is associated with probabilities P_{in} and P_{ex} such that $P_{in} + P_{ex} = 1$. The value of P_{ex} is 1/2 initially. After each round, P_{ex} is halved and P_{in} set appropriately.

This process of randomly flipping bits in V'_G is designed so that the end result tends towards the group content vector with high probability (Section 5.6). Randomization of the bit flips is used to prevent a malicious provider within the provider group from being able to determine with any certainty the value of bits in the content vector of other providers (Section 5.7).

5.4 Global Index

After the r bit-flipping rounds are complete, the vector V'_G from each provider group is sent to a designated index host. The index host receives these vectors from each privacy group along with a list of all providers in the privacy group. It then aggregates these vectors into a materialized index MI. The MI maps a bit position i to a list of providers that belong to privacy groups whose content vector has i set to 1. More formally, $MI(i) = \{p|p \in G \wedge V'_G[i] = 1\ for\ some\ privacy\ group\ G\}$.

5.5 Querying with PPI

Recall that a PPI must map each query q to a set M_q that corresponds to one of the three cases defined in Section 4.2. So far we have defined our MI mapping to map *bits* to providers, but the process of using MI as a PPI that maps *queries* to providers is straightforward: M_q is formed by first taking the conjoined terms Q specified in q and looking up each term's bit position $1 \ldots L$ in MI using the system-specified lookup (hash) function H. The provider list is formed by taking the intersection of $MI(i)$ for each such bit. More formally, $M_q = \cap_{t \in Q} MI(H(t))$. The MI thus serves as an implementation of PPI.

5.6 Correctness

We now show that the mapping PPI from a query q to provider set M_q is expected to satisfy the conditions required of a PPI. First, we show that the set of providers M_q contains all providers that share documents matching q, which is a necessary condition for cases [A] and [B] of the definition.

Lemma 5.1 *Assuming each vector V'_G used to create the mapping PPI is equivalent to or subsumes the group content vector V_G of its group G, the mapping M_q contains all providers that share a document matching q.*

The above claim follows from the simple fact that group content vectors are a logical OR of the individual content vectors from each group member. Thus, each list obtained from PPI given some term t is guaranteed to contain all providers sharing documents with the specific term. Now we establish the qualifying assumption.

Lemma 5.2 *Let c be the number of providers in a privacy group G. For any $0 < \epsilon < 1$, at the end of $r \geq max(3, -\log[1 - \{\frac{8}{7}(1-\epsilon)\}^{1/(c-1)}])$ rounds, a bit b that is 0 in V_G is also 0 in V'_G with probability $1-e^{-c}$, while a bit b that is set to 1 in V_G is also 1 in V'_G with probability $1-\epsilon$.*

Proof Let us first consider the case when $b = 0$ in V_G. This means that none of $p_1, p_2, \ldots, p_c$ will set b to 1. If b was 0 in V'_G at the start of the construction, it stays 0 until the end. If b was 1 at the start of the construction, each provider in G will attempt to reset it at each step of every round with probability p_{ex} of the round. The probability that b is still 1 at the end of r rounds is $\Pi_{i=1}^{r}(1 - 1/2^i)^c \leq e^{-c}$.

Now consider the case when $b = 1$ in V_G. Note that in the worst case, only p_1 has $b = 1$ in V_1 and the rest of p_i attempt to set b to 0. Consider the value of bit b at the start of the r^{th} round. Let b be 0 with probability P_0 and 1 with probability P_1. In the r^{th} round, $P_{ex} = 1/2^r$ and $P_{in} = 1 - P_{ex}$. The bit b is 1 at the end of round r with probability $P(b = 1) = (1 - P_{ex})^{c-1}(P_1 + P_{in}P_0) = (1 - P_{ex})^{c-1}(P_1 + (1 - 1/2^r)P_0) = (1 - P_{ex})^{c-1}(P_1 + P_0 - P_0/2^r)$. But $P_1 + P_0 = 1$ and $P_0 \leq 1$. This means $P(b = 1) \geq (1 - P_{ex})^{c-1}(1 - 1/2^r)$. For $r \geq 3$, $(1 - 1/2^r) \geq 7/8$. For any $0 < \epsilon < 1$, we can ensure that $b = 1$ with probability $1 - \epsilon$ by requiring that $P(b = 1) \geq (1 - P_{ex})^{c-1}\frac{7}{8} \geq 1 - \epsilon$ or $(1 - 1/2^r) \geq [\frac{8}{7}(1-\epsilon)]^{1/(c-1)}$ or $r \geq -\log[1 - \{\frac{8}{7}(1-\epsilon)\}^{1/(c-1)}]$.

Lemma 5.2 shows that we can make V'_G subsume the group content vector V_G with probability arbitrarily close to one by increasing the number of rounds. Henceforth, we assume that the number of rounds has been appropriately chosen so that we can safely assume subsumption. Given this assumption, we have established the following:

Theorem 5.3 *For any query q with conjoined terms Q, $M_q = \cap_{t \in Q} PPI(H(t))$ contains all providers that share documents matching q.*

All that remains to establish that the mapping M_q is expected to meet the requirements of a PPI is to demonstrate that should M_q be non-empty, then it is expected to contain at least half false positives, or be equivalent to the entire provider set.

Lemma 5.4 *Let n be the total number of providers indexed by PPI. For query q with selectivity σ, the expected number of groups that contain a provider sharing a document matching q is $\frac{n}{c} \times [1 - (1-\sigma)^c]$.*

Proof The probability that *no* provider in a group shares a document matching a query with selectivity σ is $1 - \sigma$ multiplied across all group members, or $(1-\sigma)^c$. The expected number of groups that share at least one such document is thus one minus this number multiplied by the total number of groups, or $\frac{n}{c} \times [1 - (1-\sigma)^c]$. ∎

Case [B] asks that a query q with selectivity $\sigma < 0.5$ be mapped to at least $2\sigma n$ providers. The construction ensures that should one provider in a privacy group share a document matching a query, that all other members of its group will be contained in the mapping M_q. From Lemma 5.4 we know that a query q with selectivity σ is mapped to $n[1 - (1-\sigma)^c]$ providers. Thus, Case [B] holds if $2\sigma n \leq n[1 - (1-\sigma)^c]$ or $2\sigma + (1-\sigma)^c \leq 1$. As c increases, values of σ that satisfy the equation will increase. For $c = 4$, the condition is expected to hold for $0 < \sigma \leq 0.456$, while for $c = 10$, the values are $0 < \sigma \leq 0.499$.

What if σ for a query q lies beyond this range? The definition states that the constructed index must follow (the only remaining) Case [C]. In other words, the term must map to n providers. For the constructed index, this is not true as $n[1-(1-\sigma)^c] < n$ for all c and $\sigma \neq 1$. However, we can make $n[1-(1-\sigma)^c] \rightarrow n$ by increasing c. For example, for $c = 4$, $n[1-(1-\sigma)^c] \geq 0.937n$ for $0.5 \leq \sigma \leq 1$ while $c = 10$ leads to a value of $0.999n$ for the same range of σ.

Theorem 5.5 *The mapping M_q of providers produced by PPI for any query q is expected to satisfy the conditions for being privacy-preserving.*

Note that we have not proved that the mapping M_q is *guaranteed* to be privacy-preserving. As we have quantified above, there is a small chance that for any particular query, there may not be sufficient false positives, or that the result will not quite contain the list of all providers when query selectivity is low. Nevertheless, the data points we presented above show that the probability is quite low for reasonable settings of c. With respect to any given query, then, the output of our index is *expected* to be privacy-preserving, and in fact this expectation is in practice near one for reasonable settings of c. Also note there is no telling *which* of the queries lead to output that does not precisely satisfy the definition, which limits the opportunities for an adversary to actually exploit this fact.

Lastly, we wish to emphasize that these exceptional cases are only with respect to generating a sufficient number of false positives. So long as the construction produces accurate group content vectors (which can be guaranteed with probability arbitrarily close to one), Theorem 5.3 implies correctness of the search output.

5.7 Privacy Characteristics

Let us next consider the privacy breaches that could occur during index construction. Note that the final group content vectors V'_G do not provide an adversary with information that cannot already be obtained from directly inspecting the PPI itself. This gives an adversary who is outside the provider group of another provider p_s no more information about p_s than what is available to the adversarial searcher discussed previously in Section 4.

What remains to be quantified then are the privacy breaches that occur between members of a group G while V'_G is being generated. The communication between group members consists of sending the current working vector V'_G from one provider to its successor. We assume all communications are encrypted and authenticated, and thereby immune to interception by anyone other than the intended recipient. Each member in G thus has a limited view of the construction process. Still, the following theorem shows that this limited view does leak some information that can be used for privacy breaches by an active adversary within the group.

Theorem 5.6 *Within a privacy group G, an active adversary p_a can learn that its predecessor p_s is sharing a document containing term t with probability up to* 0.71.

Proof The content vector V'_G that p_a receives could have been altered by any of the remaining $c-1$ ($c > 2$) providers but was modified by p_s most recently. The adversary p_a can remember the content vectors from each round and attempt to deduce bits being set by p_s. Trivially, p_a can deduce that none of the members have terms that result in bits $b = 0$ in V'_G at the end of r rounds. However, for p_s it can deduce more. If p_a observes a bit $b = 0$ in V'_G at the end of a round, and that bit is subsequently 1 for all remaining rounds r', then it can deduce that p_s does not have a term that sets bit $b = 0$ with probability $1 - P_{in} = P_{ex} = 1/2^{r'}$. Moreover, if it observes that a bit $b = 1$ in all the r rounds, then the probability that p_s does not have the term is $\Pi_{i=1}^{r}(1 - 1/2^i)$ which tends to the limit of 0.29 from above. In other words, p_a has deduced that p_s has a document containing the term with probability up to 0.71. ∎

More problematic is the ability with which colluding adversaries may breach privacy. In the worst case, colluding adversaries may be serving as both the predecessor and successor of a provider p_s. In such a case, the adversaries can determine precisely the bits flipped by p_s during each round. The adversaries can then determine the content vector of p_s with high probability (a privacy breach tending towards Provable Exposure).

We suggest that such attacks can be made irrelevant if a provider can ensure that neighboring providers in its group are "trustworthy". This can be achieved in practice by having providers arrange themselves along rings according to real-world trust relationships.

5.8 Efficiency

An ideal PPI is, by definition, at most $2\times$ less selective than a precise index for any given query. The index constructed by our algorithm maps queries to groups of size c. In the worst case from the perspective of efficiency (but best case from the perspective of privacy), for a given query q, only one member of each identified group actually shares a matching document. In such a case the index is $c\times$ less selective than a precise index. The worst-case loss in efficiency of the constructed index is thus proportional to the degree of privacy (c) desired by its participants. However, the expected number of providers for a query q of selectivity σ is $n[1-(1-\sigma)^c]$ by Theorem 5.4. The loss in selectivity is then $[1-(1-\sigma)^c]/\sigma$. For $\sigma = 0.1$, the loss

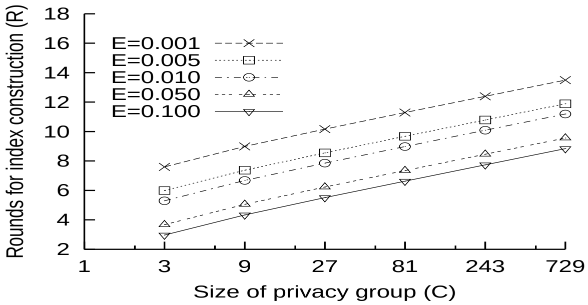

Figure 4: Number of rounds for index construction

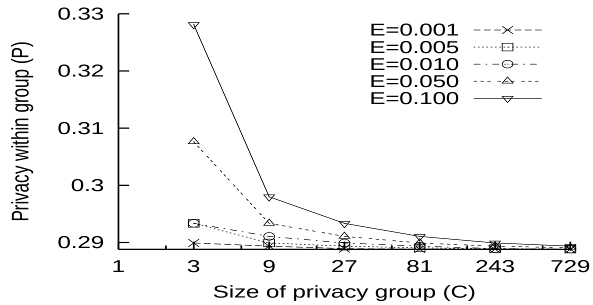

Figure 5: Loss of privacy during index construction

is 3.43 for $c = 4$ and 6.51 for $c = 10$. We discuss such expected loss in selectivity in more detail in Section 6.

Theorem 5.7 *A query q of selectivity σ has a worst-case selectivity of $c \times \sigma$ and an expected selectivity of $[1 - (1 - \sigma)^c]$ in the constructed index.*

6 Empirical Evaluation

In this section we evaluate the behavior of the analytical bounds and expectations established in the previous section, and also the performance of the indexing scheme on real data. Specifically, we show that:

- The number of rounds required by index construction is small in practice, leading to efficient index creation.
- The probability with which an adversary provider can breach privacy of its predecessor during index construction tends quickly to our bound of 0.71, which implies that careful tuning of the other problem parameters is unlikely to be useful for avoiding this problem in practice.
- On real data, reasonable settings of ϵ ensure that the generated index suffers no loss of recall.
- On real data, the performance penalty in query processing of a PPI compared to a precise provider index is roughly $\frac{2}{3} \times c$ when averaged over all queries.

6.1 Choice of Rounds

We start our evaluation of the index by studying the number of rounds required for its construction in a privacy group. As discussed in Theorem 5.2, the number of rounds r required in a group G of size c for ensuring $V_G \subseteq V'_G$ with probability $1 - \epsilon$ for $0 < \epsilon < 1$ is given by $r \geq max(3, -\log[1 - \{\frac{8}{7}(1 - \epsilon)^{1/(c-1)}\}])$. Thus, r depends on c and ϵ. We note that r is proportional to the processing and bandwidth costs incurred during index construction.

Figure 4 plots the value of r for various c and ϵ. The X-axis plots c on a logarithmic scale while the Y-axis plots r. The number of rounds r grows almost linearly with logarithm of the size c of a privacy group. Since c determines the level of privacy ensured for a member of G, index construction scales well with the privacy requirements imposed on it.

The curves in Figure 4 pull up parallel to each other for increasing ϵ values. As ϵ decreases, accuracy of the group vector increases. As can be observed, an increase in accuracy 10× (from $\epsilon = 0.100$ to $\epsilon = 0.01$ and then from $\epsilon = 0.01$ to $\epsilon = 0.001$) results in an average increase in r by a constant value of 2.5 rounds. Thus, the index construction process also scales well with desired accuracy ϵ.

6.2 Breaches within a Privacy Group

Theorem 5.6 shows that the privacy breach at a provider is the most severe for the preceding provider in the index construction chain within a privacy group. The privacy breach P_{loss} was quantified as $P_{loss} = \Pi_{i=1}^{r}(1 - 1/2^i)$. The function is plotted in Figure 5 for various values of c and ϵ. The X-axis plots the size c of a group G while Y-axis plots P_{loss}.

Figure 5 shows that the privacy breach tends to 0.29 quickly, except for small values of c and ϵ. Still, the absolute difference is quite small. This suggests that careful tuning of c and ϵ cannot be used to avoid this potential privacy breach.

6.3 Loss in Selectivity

We next study the expected increase in query processing cost incurred by the use of the constructed PPI. The increase in cost is due to the decrease in selectivity of a term in the constructed index. Theorem 5.7 implied that the expected selectivity in the constructed index for a term t with actual selectivity σ is $[1-(1-\sigma)^c]$. The loss in selectivity can be quantified as the ratio $[1 - (1 - \sigma)^c]/\sigma$. Figure 6 plots this ratio on the Y-axis against σ on the X-axis. The curves are drawn for various values of c.

We observe that as $\sigma \to 1$, the expected loss in selectivity diminishes to 1 for all curves. Indeed, we anticipate such a result as almost all providers share a document with the particular term. Both the precise and the constructed index must now map the term to all providers. The curves taper off when $\sigma nc \geq n$.

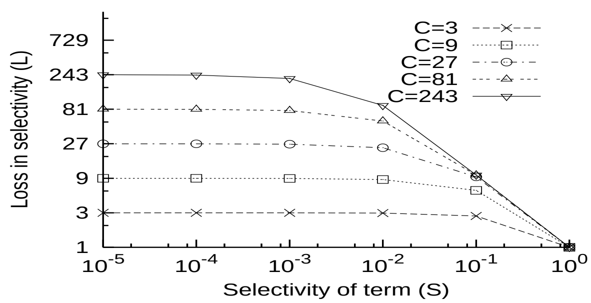

Figure 6: Loss in Selectivity

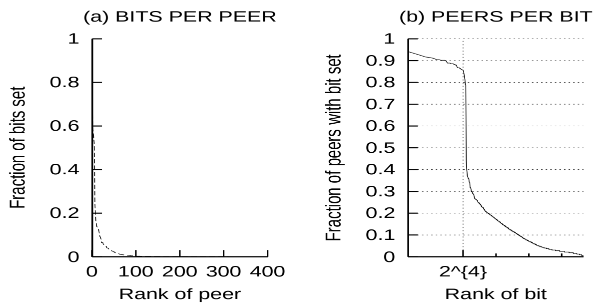

Figure 7: Characteristics of Bloom filters from 324 YouSearch peers.

The lower selectivity range is more interesting as it corresponds to rare items that are shared by a few providers. One can argue that it is the rare terms that need to be protected carefully, so that they cannot be pinned to the sharing provider. We observe here that the loss in selectivity for such terms is c. In other words, $c\times$ providers are identified as potentially having a document sharing such a term.

6.4 Characteristics of YouSearch Data

The remaining results involve running the indexing and search algorithms on a real-life dataset obtained from a deployment of YouSearch [18] within the IBM corporate intranet. A content provider in YouSearch constructs a Bloom filter content vector with length $L = 64$ Kbits. The hash function H used for constructing the Bloom filter is computed as follows. An MD5 hash is computed for each term t. The first 2 bytes are extracted from the 16 bytes long hash to be used as the value for $H(t)$. The set of terms used to create the content vector are extracted from the filenames and pathnames of shared files, and from the bodies of text and HTML file types.

The dataset used here was obtained from collecting all Bloom filters from the set of peers actively participating in the search network at a specific point in time. This dataset originally had 350 Bloom filters, though 26 were randomly discarded to make this number a multiple of 81 (used later in experiments as a candidate group size).

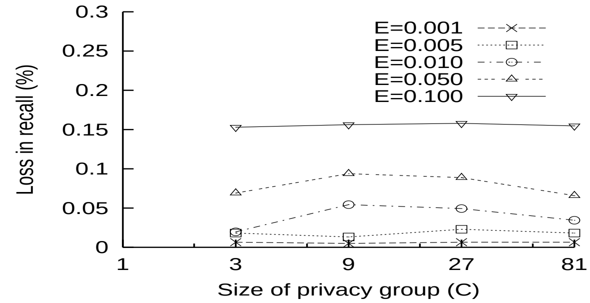

Figure 8: Loss in Recall

The characteristics of the resulting Bloom filters are depicted in Figure 7. We computed the fraction of bits set in the Bloom filter for each peer. Figure 7(a) plots the fraction of bits set in a Bloom filter on the Y-axis against peers ranked in decreasing order of such fraction on the X-axis. We observe that a few peers have a large (two-thirds) fraction of their bits set. We also computed the fraction of peers that have a bit set to 1 for each bit position of the Bloom filter. Figure 7(b) plots the fraction of peers that have a particular bit set on the Y-axis against bits ranked in decreasing order of such fraction on the X-axis. Most of the bits in the Bloom filter are highly selective, though a few bits that correspond to the most frequently occurring words (about 2^4) are set by almost 80% of peers.

6.5 Effectiveness of V'_G in Search

Recall that a choice of c and ϵ determines the magnitude of inaccuracy observed in the group content vector. We studied the effects of such inaccuracies on the *search* function. Queries asked by peers in YouSearch against shared content were logged. From a sample of $1,700$ logged queries, we randomly selected 100 distinct queries as our query set.

The peers (providers) were randomly partitioned into c sized privacy groups and indexes created for each group for various ϵ values. The hash of each term in a query was computed to determine the corresponding bit in the Bloom filters. The determined bit position was checked in the index V'_G for each group G. Groups that had bits for all terms in a query q set to 1 were deemed to be relevant. The query was then evaluated at each of the member providers. Since group indexes constructed with larger values of ϵ can be expected to have some bits falsely set to 0, the answers for q at such member providers will be lost.

For each query q, define S_q to be the set of providers that have bits for all terms in q set to 1 in their YouSearch Bloom filters. Define R_q to be the set of providers that are members of groups deemed relevant by the constructed index. Then, *loss in recall* for q due to the constructed index can be quantified as $|S_q - R_q|/|S_q|$. The average loss in recall for a query set is simply the average of loss in recall for constituent queries.

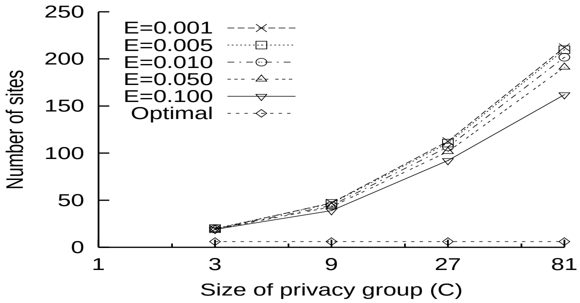

Figure 9: Query Processing Cost

Figure 8 plots size of privacy group c on X-axis against the average loss in recall observed on Y-axis. Note that the Y-axis is measured in percent (%) units. We observe that loss in recall is very small (less than 0.2%) and decreases with increasing accuracy of the index. With an ϵ setting of 0.001, the search is effectively complete.

6.6 Query Processing Cost

We next study the performance penalty in query processing of a PPI by evaluating the sample query set against the YouSearch dataset. Figure 9 plots the processing cost incurred averaged over all queries. The X-axis plots size of privacy group c while Y-axis plots the average number of sites that evaluate a query. In the absence of privacy concerns, the query would be evaluated only at sites that had the corresponding bit positions in their Bloom filters set to 1 (indicated as the *Optimal* curve). However, the index causes *all* c sites in a c sized privacy group to be searched. We observe that increasing c leads to a concomitant increase in query processing costs. When averaged over all queries in our test set, the performance penalty is roughly $\frac{2}{3} \times c$. An increase in ϵ (decrease in accuracy) also results in decreased processing cost. However, such decreases are merely an artifact of reduced recall due to the group content vector inaccuracies implied by larger settings of ϵ.

7 Related Work

Researchers have identified the importance of preserving privacy during searching. The celebrated paper by Chor et. al. [5] introduced the problem of *Private Information Retrieval*. A user wishes to privately retrieve the i-th bit from a database, without revealing any information about i. Gertner et. al. [13] introduced the stronger model of *Symmetrically Private Information Retrieval* which, in addition to maintaining the user's privacy, prevents the user from obtaining any information other than a single bit of the data. The privacy of the user is defined in an information-theoretic setting, which makes it hard to find practical and efficient schemes. SPIR schemes often require multiple non-colluding servers, consume large amounts of bandwidth and do not support keyword searching. Although our scheme does not provide information-theoretic security bounds, it has low computational and communication complexity while providing probabilistic privacy guarantees with keyword searching.

Statistical database research strives to provide aggregate information without compromising sensitive information about individual records [1]. Secure databases research attempts to prevent unauthorized access to records in the database [20]. Popular search solutions build inverted indexes on shared content [4, 16, 22]. All of these solutions assume a trusted centralized server. We believe that autonomous content providers will find it immensely difficult to form a consensus on such a trusted host.

Researchers have investigated the problem of running queries over encrypted data at an untrusted server [17, 25]. The schemes require the searcher to know a secret key with which content accessible to the searcher is encrypted. The searcher now has to explicitly maintain secret keys for each provider she has access to. As far as we know, no previous work has defined PPI to enable global keyword searches.

In practice, people have preferred replacing privacy definitions with *anonymity* where the requirement is that the identity of the user ("Bob") be masked from an adversary ("Alice"). Low cost systems have been designed for various applications that involve building an "anonymous channel" that hides Bob from Alice. For example, Onion Routing [27], Crowds [23] and Tarzan [10] allow the source of a message to remain anonymous. Freenet [11] and FreeHaven [8] ensure provider privacy for file sharing. Freenet supports searches but cannot support access control over shared content. Freehaven does not support searches but preserves the anonymity of readers. It is not obvious how these schemes can be adapted to enable content privacy while searching access-controlled content.

Researchers in the field of secure multi-party computation [15] have developed theoretical methods for securely computing functions over private information distributed across any number of hosts. Recent work in this area has focused on developing more efficient schemes for specific functions. Development of a secure yet practical multi-party algorithm for computing the OR of distributed bit vectors could be useful for improving the privacy of our PPI construction scheme. Secure co-processors [9] could also be used for improving the privacy of PPI construction, should one with sufficient processing power, storage and bandwidth be available for use by participating providers.

8 Conclusions

We have addressed the challenge of providing privacy-preserving search over distributed access-controlled content. Conventional inverted indexes represent an indexed document in its virtual entirety. The trust and security thus required of any host providing such an index over access-controlled content is enormous. In fact, as the number of participating information providers grows, this required level of trust quickly becomes impractical. Our solution eliminates entirely the need for such a trusted indexing host through the use of a *privacy-preserving index (PPI)*.

We defined and analyzed a PPI and presented a randomized algorithm for constructing a PPI. We showed that the index, once constructed is strongly resilient to privacy breaches even against *colluding* adversaries. The construction process allows only a limited privacy loss and is susceptible to colluding adversaries. However, we argued that such conditions can be minimized through careful arrangement of providers in the construction process based on real-world trust relationships. Experiments on a real-life dataset validate performance of our scheme.

Our solution enables information providers to maintain complete control in defining access groups over their content and ensuring its compliance. Moreover, system implementors can use the size c of privacy groups as a tuning knob to balance privacy and efficiency concerns for their particular domains.

References

[1] N. R. Adam and J. C. Wortman. Security-control methods for statistical databases. *ACM Computing Surveys*, 21(4), 1989.

[2] M. Bawa, R. J. Bayardo Jr., S. Rajagopalan, and E. J. Shekita. Make it fresh, make it quick - searching a networks of personal webservers. In *Proc. of the Conf. on World Wide Web (WWW)*, 2003.

[3] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of ACM*, 13(7):422–426, 1970.

[4] S. Brin and L. Page. Anatomy of a large-scale hypertextual web search engine. In *Proc. of the Conf. on World Wide Web (WWW)*, 1998.

[5] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. of the Conf. on Foundations of Computer Science (FOCS)*, 1995.

[6] A. Choudhury, N. Maxemchuk, S. Paul, and H. Schulzrinne. Copyright protection for electronic publishing over computer networks. *AT&T Bell Laboratories Technical Report*, 1994.

[7] T. Dierks and C. Allen. The tls protocol. *RFC 2246, Standards Track, Network Working Group*, 1999.

[8] R. Dingledine, M. J. Freedman, and D. Molnar. The free haven project: Distributed anonymous storage service. In *Proc. of the Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[9] J. Dyer, M. Lindemann, R. Perez, R. Sailer, S. W. Smith, L. van Doorn, and S. Weingart. Building the ibm 4758 secure coprocessor. *IEEE Computer*, 34, 2001.

[10] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proc. of the ACM Conf. on Computer and Communications Security*, 2002.

[11] The freenet project (http://freenetproject.org).

[12] A. Frier, P. Karlton, and P. Kocher. The ssl 3.0 protocol. *Netscape Communications Corp.*, 1996.

[13] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *Proc. of the ACM Symposium on Theory of Computation (STOC)*, 1998.

[14] The gnutella network (http://gnutella.com).

[15] S. Goldwasser. Multi-party computations: Past and present. In *Proc. of the ACM Symp. on Principles of Distributed Computing*, 1997.

[16] L. Gravano, H. Garcia-Molina, and A. Tomasic. Gloss: Text source discovery over the internet. *ACM Transactions of Database Systems*, 24(2), 1999.

[17] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *Proc. of the (ACM) Conf. on Management of Data (SIGMOD)*, 2002.

[18] R. J. Bayardo Jr., A. Somani, D. Gruhl, and R. Agrawal. Youserv: A web hosting and content sharing tool for the masses. In *Proc. of the Conf. on World Wide Web (WWW)*, 2002.

[19] The kazaa media network (http://www.kazaa.com).

[20] C. Landwehr. Formal models of computer security. *ACM Computing Surveys*, 13(3), 1981.

[21] G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *Proc. of the Symposium on Internet Technologies and Systems (USITS)*, 2003.

[22] Napster file-sharing (http://www.napster.com).

[23] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[24] J. Ritter. Why gnutella can't scale. no, really. 2001.

[25] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proc. of the (IEEE) Symposium on Security and Privacy*, 2000.

[26] I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of ACM SIGCOMM*, pages 149–160, 2001.

[27] P. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *Proc. of the IEEE Symposium on Security and Privacy*, 1997.

[28] C. Tang, Z. Xu, and M. Mahalingam. Peersearch: Efficient information retrieval in peer-to-peer networks. In *Proc. of the HotNets-I*, 2002.