Providing Privacy through Plausibly Deniable Search*

Mummoorthy Murugesan† Chris Clifton†

Abstract

Query-based web search is an integral part of many people's daily activities. Most do not realize that their search history can be used to identify them (and their interests). In July 2006, AOL released an anonymized search query log of some 600K randomly selected users. While valuable as a research tool, the anonymization was insufficient: individuals were identified from the contents of the queries alone [2]. Government requests for such logs increases the concern. To address this problem, we propose a client-centered approach of *plausibly deniable search*. Each user query is substituted with a standard, closely-related query intended to fetch the desired results. In addition, a set of k-1 cover queries are issued; these have characteristics similar to the standard query but on unrelated topics. The system ensures that any of these k queries will produce the same set of k queries, giving k possible topics the user could have been searching for. We use a Latent Semantic Indexing (LSI) based approach to generate queries, and evaluate on the DMOZ [10] webpage collection to show effectiveness of the proposed approach.

1 Introduction

Search engines such as Google, Yahoo!, and MSN boast huge user bases. Query logs can give extensive insight into people's interests and activities. While valuable in aggregate, this data can also be used to develop profiles of individuals, raising privacy concerns.

Anonymizing the log metadata does not solve this issue. In July 2006, AOL released an anonymized search query log of around 600,000 randomly selected users. The logs had been anonymized (at the server side) by removing individually identifying information such as IP address, username, and any other personal information associated with that user. This left a log containing the actual query, timestamp, clicked URL, and a random ID assigned for each user. However, this simple anonymization proved ineffective; the *query itself* often contained identifying information[2] (e.g., ego-surfing). It has been shown that very accurate classifiers can be built to identify the gender, age, and location of users [16] based on query text.

Adding to this concern is recent government subpoenas for search logs [13]. While not requesting identifying information, the government did request the query text - which as AOL discovered, may be inherently identifying. This can have serious implications both for individuals and for the search engine companies. There are two problems; the ability of the search engine to identify the user (identifiability) and the ability to link a query to the user (linkability). Other (complementary) work addresses identifiability through metadata (e.g., IP address); this is discussed in Section 6. We focus primarily on linkability (although as we shall see, we also address identifiability of the query.)

Most of the work in this area concentrates on anonymizing the log metadata (either at the server side, or under client control using anonymous routing techniques); we are aware of only one that addresses protecting search text [14]. We discuss this and other related work in Section 6.

1.1 Plausibly Deniable Search.

We address this problem at the client side by generating a set of cover queries for each real query. This eliminates timing-based attacks to discover the real query (a problem with [14]). While not completely hiding the real query, if done properly it supports a notion of *plausible deniability*. While it is possible that an individual may have issued the actual query, it is equally plausible that they issued one of the generated cover queries. This places a burden of proof on the user of the query log when trying to imply something about an individual's interests.

We introduced the idea of Plausibly Deniable Search in [19]. More formally, assume a server log has a set of queries $S = \{Q_1, \ldots, Q_k\}$ from a particular user at a particular timestamp. A party with access to this log tries to prove that the actual search was Q_i. K-plausible deniability holds if the user can prove that any $Q_j \in S$ is as likely to have been the actual query:

*This material is based upon work supported by the National Science Foundation under Grant No. 0428168. Partial support for this work was provided by MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research.

†Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA, **{mmuruges,clifton}@cs.purdue.edu**

DEFINITION 1.1. *(PD-Privacy) A user issuing a set of queries, $S = \{Q_1, \ldots, Q_k\}$, where $Q_i \in S$ is the desired query, has* k-Plausibly Deniable privacy *of Q_i if*

1. *the user can show that any query $Q_j \in S$, would have generated the set S with equal probability,*
2. *all $Q_j \in S$ are on different topics, and*
3. *all $Q_j \in S$ are equally plausible as actual user queries.*

Constraint 1 is relatively straightforward. Constraint 2 ensures that the cover queries mask the user's intent, not simply the words used. The final constraint is perhaps the most difficult – cover queries that are randomly selected from a predefined list, while including an arbitrary user query Q_i, would not meet this test. Nor would a random selection of words (unless the user happens to issue a random set of words as their query.)

For example, assume a user issues a query 'galaxy sun' and this scheme generates a cover query 'house garage'. Both queries are plausible and appear completely unrelated. If both queries are submitted to the server, the user can filter the documents based on the original query. A third party having access to the query log, 'galaxy sun' and 'house garage', claims that the user searched for 'galaxy sun'. Given that these queries satisfy *PD-Privacy*, in the absence of other evidence that the user is interested in astronomy, the user can make a valid claim to have actually been investigating construction. Notice that we are more concerned about the linkability of queries to the user than the user identification; this is further discussed in Section 1.2.

We propose a Latent Semantic Indexing (LSI) based semantic mapping to generate a fixed set of possible queries, avoiding the problem of identifying the real query as one that wouldn't be generated by the method, and fixed sets of cover queries to meet constraint 1. We cluster to generate sets that meet constraints 2 and 3, ensuring that cover queries are qualitatively similar but on different topics. The results are filtered at the client using the original query, thus (for the most part) hiding the effect from the user. This will be covered formally in Algorithm 3, after we have described the components used to accomplish Plausibly Deniable Search.

We next discuss types of web search privacy, then formalize the definition of plausibly deniable search in Section 2. We detail the approach in Section 3, followed with an experimental analysis on DMOZ webpage collection in Section 5. A sufficiently large query log could be analyzed for sequences of related queries to distinguish actual from cover queries; we analyze the effectiveness against such attack in Section 5. We discuss related work and other approaches in Section 6.

1.2 What Privacy does this Provide? Search engines store the IP address of each search request, a unique SessionID, timestamp, and in some cases more detailed information (user identity across sessions collected through stored cookies, etc.) as the identifying information of the user. This leads to two classes of privacy concerns:

Identifiability - Who issued a query?
If the individual who issued a query cannot be determined, privacy is generally not an issue. The ability to identify the true IP address is often enough for identification purposes. Several techniques exist to mask IP address, such as onion routing [11]; A more comprehensive solution, eliminating tracking via browser- and machine-specific information, has been released with Private Web Search[21]. However, the AOL query log release incident has proven that queries themselves could be used to identify the individual.

Linkability - What was issued?
Simply knowing that an individual has issued a query is rarely a privacy concern, unless we link that individual to *what the query is about*. Linkability is about determining the interests of an individual from the query. The need for considering linkability arises because 1) many users find anonymization networks slow and inconvenient, 2) some of these anonymization schemes could be attacked [3] to reveal user identity, and 3) the queries may inherently reveal identity. Linkability protection hides the knowledge of what the individual is searching for even if identity is known. Private Information Retrieval[8] is an example of a pure linkability solution, although the requirement for (computationally expensive and revenue-model-destroying) server-side cooperation make it infeasible for practical web search.

Plausibly Deniable Search formally addresses only linkability, although we shall see that the technique presented also protects against identifiability through the query text. Note that the clicked urls immediately reveal the user query. In PDS, we assume that the client hides clickthrough information from the search engine, using techniques such as in [21]. Combining with identifiability protection techniques such as Private Web Search can provide a full query privacy package.

2 Defining Plausibly Deniable Search

We refer to a set of queries satisfying Definition 1.1 as a *PD-Queryset*. A user who wishes PD-Privacy finds (or generates) the PD-Queryset containing the closest query to the desired query, and issues the corresponding PD-Queryset. Our goal is to find these PD-Querysets (through either a static or a dynamic process) so that for any user query: 1) the query belongs to a PD-Queryset, or there is a PD-Queryset returning similar results; and

2) the queries in the set are similar in strength in their respective topics. By strength, we mean how focused each query is to the topic it retrieves; a tightly focused query should be masked with similarly tight queries on different topics; a random set of words should be masked with queries consisting of (different) random word sets.

The rest of this section gives a definition of the problem that allows us to discuss the quality of a solution, as well as definitions specific to the particular solution approach presented in Section 3.

We define a universal term set $W = \{w_1, \ldots, w_n\}$, and a universal set of queries Q as the power set of W.

DEFINITION 2.1. *(Universal Queryset Q) Let W be the set of all possible terms. The Universal Queryset is the power set of W, i.e.,* $Q = P(W) = \{X : X \subseteq W\}$.

Note that Q includes every possible query, even of lengths varying from 1 to $|W|$. It is unusual for a query to have more than 10 terms. We define Q in this way to be fully general in the definition.

To show a cover query hides user intent, we must have a way to judge the intent of a query. We do this by assuming that each query in Q is associated with a set of topics. For example, a query {java} could belong to topics programming languages and coffee. The topics themselves could be related by one being the general of the other; programming languages is covered by a more general topic computer science.

Let T be the set of all topics and *topics* be a function that returns the topics of query q such that $topics(q) \subseteq T$. For each topic $t \in topics(q)$, we define the query-topic relationship score as $rscore(q, t)$.

DEFINITION 2.2. *(Query-Topic Score) Given a query q and a topic t, the query-topic relationship score (rscore) function returns the measure of relationship between q and t:*

$$rscore : Q \times T \longmapsto R$$

If $rscore(q_1, t_1) > rscore(q_1, t_2)$ then the query q_1 belongs to topic t_1 more than t_2. In this definition, we do not attempt to define a specific scoring function for the query-topic relationship. There could be different scoring schemes, based on a specific ontology for the query terms or pre-defined categories. The $rscore(q, t)$ just captures the magnitude of the relationship coming out of such scoring schemes. The *dominating topic* of q is defined as the topic with the highest $rscore(q, t)$. The $dom(q)$ is the topic t_c such that $rscore(q, t_c)$ is $\max(rscore(q, t_i)), \forall t_i \in T$.

We now define the topic vector of a query q as a $|T|$ dimensional vector $\vec{q^t} = \langle rscore(q, t_1), \cdots, rscore(q, t_{|T|})\rangle$. The $|T|$-dimensional topic vector of q contains $rscore(q, t_i)$ for each topic $t_i \in T$. Using the topic vectors, we can compare queries with respect to their topics.

DEFINITION 2.3. *(Query Topic Comparison) Let C be the query topic comparison function, defined as* $C : Q \times Q \longmapsto R$. *The comparison function of two queries* (q_1, q_2) *is the cosine similarity of their topic vectors* $\vec{q_1^t}$ *and* $\vec{q_2^t}$*:*

$$C(q_1, q_2) = \frac{\vec{q_1^t} \bullet \vec{q_2^t}}{||\vec{q_1^t}||\ ||\vec{q_2^t}||}$$

The function C compares the queries by computing the cosine of the angle between their topic vectors.

The definitions above give us the ability to compare queries, and also define constructs on their topics. The query-topic relationship score (*rscore*) gives a measure of how a particular query is related to a given topic. In these definitions, we assume that the list of topics is available, which may not be always realistic. In later sections we will present a query-topic comparison function, implemented efficiently using Latent Semantic Indexing (LSI). We now formally define the plausibly deniable search as follows:

DEFINITION 2.4. *(k-Plausibly Deniable Search) Let* $q_u \in Q$ *be the user issued query. The plausibly deniable search system (PDS) produces a set of k queries* $PDS(q_u) = \{q_1, \cdots, q_k\}$ *as* Q_{submit} *with the following properties:*

- *Coverage: There exists a query* $q_i \in Q_{submit}$ *such that* $C(q_u, q_i) \geq \delta_c$. *This condition ensures that one of the queries in* Q_{submit} *will fetch the results of* q_u*, since their topic vectors have high similarity.*
- *Deniability: For every query* $q_i \in Q_{submit}$*, there exists a user query* q'_u *such that* $PDS(q'_u)$ *will be same as* Q_{submit}. *This property enables the user to deny issuing* q_u *because there are* at least $k-1$ *other queries that could result in* Q_{submit}.
- *Diversity: Deniability has no value if the cover queries also disclose user interests. This condition is written as* $C(q_1, q_2) \leq \delta_d, \forall q_1, q_2 \in Q_{submit}$ *when* $q_1 \neq q_2$. *This guarantees that any pair of distinct queries have dissimilar topic vectors, hence their topic diversity.*
- *Plausibility: All queries in* Q_{submit} *are equally plausible. We require each query to be related to its dominating topic equally:* $\delta_l < rscore(q_i, dom(q_i)) < \delta_h$.

Given a user query q_u, a PDS system generates $k-1$ cover queries that are of diverse topics and have similar

relationships/strengths to their respective topics. By the *deniability* argument, privacy is protected since each of k queries could be the original user query.

2.1 Sequentially Edited Queries and Personal Queries. In web search, many queries are *sequentially edited*; the user submits an initial query, and the next few queries are refinements of the initial query. As defined, a PDS could fail to provide deniability, since the sequence of user queries may not be masked by a coherent sequence of cover queries. Given space limitations, we leave formal definition of PDS over sequential queries as future work. However, informally we propose that a PDS system should maintain a history, and where a query shares similarities with recent queries, generate cover queries that are equivalently similar to recent cover queries. Plausible deniability would then be based on the ability to generate the sequence of queries and cover queries from any of the initial queries. We give an informal analysis of our system's resistance to such attack in Section 5.5.

"Ego-surfing" for personal information such as one's own name, SSN, credit card numbers, etc. leads to query identifiability. While a cover query could contain similar identifiable information (e.g., John Smith, with a cover query of Jane Doe); a more likely scenario is that the query would be too specific to plausibly cover. If this happens, the PDS system can inform the user, who can decide whether to risk privacy by issuing a query without plausible deniability.

3 Construction of PD-Querysets

We now present a method for statically constructing PD-Querysets based on a set of seed documents. The idea is that we extract terms from the documents to form a (large) set of possible queries. Each term set forms a *canonical query*; only canonical queries are sent to the search engine. This prevents discovery of the actual user query by distinguishing it from ones that could be generated as cover queries.

We use singular value decomposition (SVD) [12] to "map" these terms in a semantic space. We present a novel approach to cluster canonical queries based on position and characteristics of this SVD map to form PD-Querysets that are diverse in topics, yet similar in relative strength and coherence (e.g., "java programming" and "apple cider" are topically diverse but quite coherent; "java" and "rock" are topically diverse and each is closely tied to two different topics.) Given an actual user query, we also find the closest canonical query using this SVD; the corresponding PD-Queryset is issued to the server and the results of the canonical query filtered at the client using the actual user query. We now run through this process in more detail.

3.1 Singular Value Decomposition. The use of Singular Value Decomposition to map terms in a topic space forms the basis of Latent Semantic Indexing (LSI) [9]. LSI addresses the problems of polysemy and synonymy by constructing the term-document relationships in the semantic space. Even if terms are different lexically, if they share a similar meaning, they tend to be close in the semantic space.

Let the term-document matrix A be an $n \times m$ matrix, where n is the number of terms, and m is the number of documents. Let a_{ij} in A represent the frequency of term i in document j. The Singular Value Decomposition [12] of A gives us three matrices such that

$$A = USV^T, \tag{3.1}$$

where U is an $n \times n$ left singular matrix, S is an $n \times m$ diagonal matrix[1], and V is an $m \times m$ right singular matrix. The diagonal elements in S are the singular values, and are in descending order. If $rank(A) = r$, then the diagonal matrix S has exactly r non-zero singular values. Dimensionality reduction on A can be done by keeping only $r < min(m, n)$ diagonal values. The first r dimensions are the significant factors that describe the matrix A.

3.1.1 Semantic Space. Let U_r represent the $n \times r$ matrix that is created from U by keeping only r columns. Similarly the $r \times r$ matrix S_r, and $r \times m$ matrix V_r^T are generated. Thus, the term vectors of U has only r components in U_r. The basis of LSI is the notion that the reduced r-dimensional space preserves the semantics of terms. Any query term vector $\vec{q}$ can be mapped to the semantic space by the following equation:

$$\vec{q'} = q^T U_r S_r^{-1}, \tag{3.2}$$

where $\vec{q'}$ is the vector representation of q in the concept space. (For the remainder of the paper, we will use this vector notation to refer to the semantic space mapping of a query; query terms are given without the $\vec{}$.) The multiplication by S_r^{-1} differentially weights each concept dimension. The $q^T U_r$ alone gives us the unweighted query vectors as the sum of the individual term vectors of q.

3.2 Seed Terms and Canonical Queries. To ensure a reversible mapping of queries to cover queries

[1] Formally, the diagonal of the square matrix forming the upper left corner of S is non-zero and all other entries are zero.

(i.e., if the user issued a query mapping to one of the cover queries, the PD-Queryset would be the same), we predefine the PD-Querysets. While dynamically generating such clusters is an interesting avenue for exploration, the current paper does not address this issue. Unfortunately, this means it is infeasible to represent all (or even a fraction) of the exponential number of possible queries in $|W|$. Instead, we generate the canonical (and cover) queries from single terms and term pairs that occur frequently in the document collection. Since our goal is reasonable recall (precision can be obtained by filtering at the client using the original query), we believe using high frequency terms is appropriate.

We start with single terms and pairs of terms that occur at least Δ times in the seed document collection. Using frequent pattern mining on the term-document matrix with support Δ and the stopping level as 2, we extract frequent single-term and term pairs as seed queries. These seeds are mapped into the semantic space (Equation 3.2). These terms and term pairs are usually insufficient to represent actual user intent. Therefore, we generate canonical queries by grouping semantically similar seed queries. As shown in Algorithm 1, we use the KDtree data structure to find nearest neighbors. Each seed query point, with its nearest points in the semantic space, forms a group. Combining the seed terms in a single group gives canonical queries with length from two to six terms.

Algorithm 1 Canonical Queries Generation

Require: Set of seed query points(S) in the semantic space with r dimensions

1: Create a KDtree and insert all the points from S into it.
2: $Q_C \Leftarrow \emptyset$
3: **for all** $s_i \in S$ **do**
4: Find two closest neighbors - c_1,c_2 of s_i using the KDtree
5: $cquery = s_i \cup c_1 \cup c_2$
6: **if** $cquery$ is not already added to Q_C **then**
7: $Q_C = Q_C \cup cquery$;
8: **end if**
9: **end for**
10: Return Q_C as the set of canonical queries

Since our main aim is to hide the actual user query (q_u), it needs to be suppressed. However, we have the problem of recall: the search query should fetch relevant documents to q_u. This is accomplished by always issuing standard queries (i.e. canonical queries) that are semantically close to user queries. The canonical queries contain sufficient specificity to act as a reasonable surrogate for the actual user query. The complexity of this step is bounded by $O(N \log\ N)$ where N equals $|S|$, the number of seed queries.

3.3 Construction of PD-Querysets (ACP). To form a PD-Queryset, we need to group canonical queries (Q_C) into sets that are topically diverse and yet equally plausible. To do this, we create a set of measures that capture both diversity and plausibility. Diversity can be captured through distance in the semantic space. Plausibility is more difficult; some combinations of terms would seem quite implausible to a person. To support a human comparison of plausibility, we assume access to a reasonably large query log $Q_L = \{q : q \in Q\}$. (Note that Q_L and D can be independent). In addition to the mapping of Q_C into the semantic space, we also map the queries in Q_L into the semantic space. We can then use the relative density of Q_L in the neighborhood of a canonical query (in the semantic space) to judge relative plausibility. Thus canonical queries that are closely related to many "real" queries will be in the same PD-querysets, and likewise implausible canonical queries (not related to many real queries) will be brought together. This supports our desire that the canonical query corresponding to the user's intent and the cover queries be equally plausible (or implausible.)

3.3.1 Similarity Function. We define a (dis)similarity function between canonical queries as a sum of following three components:

1. Euclidean Distance in Semantic Space: Let $\vec{q}_1$ and $\vec{q}_2$ be query vectors in the semantic space. The Euclidean distance between them is as follows:

$$edist(\vec{q}_1, \vec{q}_2) \quad = \quad \sqrt{\sum_i \left(\vec{q}_1[i] - \vec{q}_2[i]\right)^2}$$

   Presumably, queries with a high Euclidean distance in the semantic space are on different topics.

2. Magnitude: The magnitude of a vector $\vec{q}$ is,

$$||\vec{q}|| \quad = \quad \sqrt{\sum_i \vec{q}[i]^2}$$

   The magnitude of a query vector defines the strength of the query in the semantic space, similar to the $rscore$ (Definition 2.2) in Section 2.

3. Neighborhood Count: Having a very dense semantic-neighborhood means that the query's meaning has more popularity (i.e., more queries being mapped to that region); hence a high commonality score. However, a query with a sparse neighborhood increases the oddity score. The neighbor

is constructed through queries from the query log (Q_L) containing real user queries. Let $\vec{\delta}$ be the difference vector to construct a imaginary neighborhood of hypercube around $\vec{q}$.

$$nhc(\vec{q}) \quad = \quad count(\vec{q}, Q_L, HCUBE(\vec{q}, \vec{\delta})$$

HCUBE is a hypercube constructed by the coordinates $\vec{q}[i] \pm \vec{\delta}[i]$. The neighborhood count function $nhc()$ takes $\vec{q}$, a query log Q_L, and $\vec{\delta}$, and returns the number of query vectors from Q_L inside the hypercube.

The dissimilarity between query vectors is defined as:

$$\begin{aligned} dis(\vec{q_1}, \vec{q_2}) \quad &= \quad (1 - edist(\vec{q_1}, \vec{q_2})/\alpha) + |\ ||\vec{q_1}|| - ||\vec{q_2}||\ |/\beta \\ &\quad + |nhc(\vec{q_1}) - nhc(\vec{q_2})|/\gamma \end{aligned} \tag{3.3}$$

where,

$$\begin{aligned} \alpha \quad &= \quad MAX(edist(\vec{q_i}, \vec{q_j})), \forall i, j \\ \beta \quad &= \quad MAX(||\vec{q_i}||) - MIN(||\vec{q_j}||), \forall i, \forall j \\ \gamma \quad &= \quad MAX(|nhc(\vec{q_i}) - nhc(\vec{q_j})|), \forall i, j \end{aligned}$$

α is the maximum Euclidean distance between all pairs of queries in Q_C. β is the largest difference in magnitudes among all pairs of queries. γ is the maximum possible difference in neighborhood count between all pairs of queries in Q_C. Thus, queries with similar neighborhood, similar magnitude, but far apart in the semantic space get a low dissimilarity score and should be placed in the same PD-Queryset.[2]

Now, we define the dissimilarity metric between a set of queries. Given two sets of queries, $A = \{\vec{a_1}, \cdots, \vec{a_n}\}$ and $B = \{\vec{b_1}, \cdots, \vec{b_m}\}$, the dissimilarity between them is calculated as follows:

$$\begin{aligned} dis(A, B) \quad &= \quad (1 - \alpha_1/\alpha) + \beta_1/\beta + \gamma_1/\gamma \\ where, \alpha_1 \quad &= \quad MIN(edist(\vec{a_i}, \vec{b_j})), \forall i, j \\ \beta_1 \quad &= \quad \left| (\sum_{i=1}^{n} ||\vec{a_i}||)/n - (\sum_{j=1}^{m} ||\vec{b_j}||)/m \right| \\ \gamma_1 \quad &= \quad \left| (\sum_{i=1}^{n} nhc(\vec{a_i}))/n - (\sum_{j=1}^{m} nhc(\vec{b_j}))/m \right| \end{aligned} \tag{3.4}$$

In computing the dissimilarity between the clusters of queries, we use the average vector length and the average neighborhood count of the individual clusters. For the Euclidean distance, we select the minimum Euclidean distance between any pair of queries from the two clusters. This condition helps prevent two queries from the same topic being put in a single cluster.

3.3.2 Agglomerative Clustering of PD-Querysets (ACP). We present an agglomerative clustering algorithm that produces the PD-Querysets, given a list of canonical query vectors Q_C and a deniability parameter k. As shown in Algorithm 2, we first construct Level-1 sets, each containing pairs of queries. We compute a dissimilarity matrix between all pairs of queries in Q_C. From the dissimilarity matrix, we select the query pairs in descending order of dissimilarities and add them to L_1. For some query pair $(\vec{q_i}, \vec{q_j})$ with the dissimilarity d_1, either q_i or q_j could have been part of a better match with dissimilarity $d < d_1$. If so, then $\vec{q_i}$ or $\vec{q_j}$ must be already in L_1, and the next pair with the lowest dissimilarity is selected. Having constructed L_1, the next level sets (L_2, L_3, etc.) are produced my merging the sets from one level below. We also ensure that the resultant sets contain non-overlapping cluster of queries. This merging process continues until the level number l equals $\log_2 k$, at which the size of the query cluster is k (2^l). Given that we need at least k queries in the resulting cluster, we have the condition that $2^l \geq k$, giving the stopping criteria for the merging procedure as $l < \log_2 k$.

Algorithm 2 The ACP Algorithm

Require: $Q_C = (\vec{q_1}, \ldots, \vec{q_n}), k$; k is the privacy parameter; Query log Q_L.

1: Compute the $n \times n$ dissimilarity matrix using the function $dis(\vec{q_1}, \vec{q_2})$;
2: $L_1 \Leftarrow \emptyset$
3: **while** $|L_1| < n/2$ **do**
4: Get query pair $(\vec{q_i},\vec{q_j})$ of queries that has the next lowest dissimilarity
5: **if** $\vec{q_i}$ or $\vec{q_j}$ is not already added to L_1 **then**
6: Create a cluster $C = \{\vec{q_i}, \vec{q_j}\}$, and $L_1 = L_1 \cup C$;
7: **end if**
8: **end while**{L_1 contains the first level pairs}
9: $l \Leftarrow 1$
10: **while** $l < \lceil \log_2 k \rceil$ **do**
11: $L_{l+1} \Leftarrow \emptyset$
12: **for all** $C_i \in L_l$ **do**
13: Find $C_j \in L_l$ such that $dis(C_i, C_j)$ is minimum
14: $L_{l+1} \Leftarrow L_{l+1} \cup Cluster(\{C_i \cup C_j\})$
15: $L_l \Leftarrow L_l - C_i - C_j$;
16: **end for**
17: $l \Leftarrow l + 1$;
18: **end while**{L_l contains the final PD-Querysets}

[2] We have chosen to equally weight the components for this early work. While weights to achieve topical diversity could be optimized using existing datasets, balancing with plausibility will demand extensive user studies.

3.3.3 Efficient similarity calculations Equation 3.3 is simple enough to be calculated easily once the values of α, β, and γ are known. However, the computation of Equation 3.4 is made more efficient using the following global data structures:

- Normalized Magnitude : For all $\vec{q_i} \in Q_C$, calculate the magnitude and store $||\vec{q_i}||/\beta$ it in $nmag[i]$.
- Normalized Neighborhood Count: For all $\vec{q_i} \in Q_C$, the normalized neighborhood count $nhc(\vec{q_i})/\gamma$ is stored in a table $nnhc$ and is accessible as $nnhc[i]$.
- Normalized Euclidean distance(i,j): The Euclidean distance $n \times n$ matrix is calculated, and $nedist[i,j]$ gives the normalized Euclidean distance between $\vec{q_1}$ and $\vec{q_2}$ as $1 - (edist(\vec{q_1}, \vec{q_2})/\alpha)$.

Equation 3.3 is now computed as,

$$\begin{aligned} dis(\vec{q_i}, \vec{q_j}) &= nedist[i,j] + |nmag[i] - nmag[j]| \\ &\quad + |nnhc[i] - nnhc[j]| \end{aligned}$$

If we store the sum of average normalized magnitude and neighborhood counts while constructing a cluster, dissimilarities between clusters can be calculated efficiently. Suppose we have a cluster $C = \{\{i,j\}, v\}$, where v is the sum of average normalized magnitude and neighborhood counts of $\vec{q_i}$, and $\vec{q_j}$, i.e., $v = (nmag[i] + nmag[j])/2 + (nnhc[i] + nnhc[j])/2$. With this representation of C, adding a query $\vec{q_k}$ results in a cluster $C' = \{\{i,j,k\}, v'\}$ where $v' = (2v + nmag[k] + nnhc[k])/3$. Similarly, by using v the dissimilarity between C and $\vec{q_k}$ is calculated as $dis(C, \vec{q_k}) = |v - mag[k] - nnhc[k]| + MAX(nedist[i,k], nedist[j,k])$. This can be generalized to find the dissimilarity between any two clusters:

DEFINITION 3.1. *Let $C_1 = \{I_1, v_1\}$, and $C_2 = \{I_2, v_2\}$ be two clusters of query vectors. I_1 and I_2 contain indices of queries contained in respective clusters. v_1 and v_2 are the sum of average normalized magnitude and neighborhood counts of queries in I_1, and I_2 respectively.*

The dissimilarity between the clusters C_1 and C_2 is:

$$dis(C_1, C_2) = v_1 - v_2 + MAX_{\forall i \in I_1, j \in I_2}(nedist[i,j])$$

The merge operation on two clusters C_1 and C_2 is:

$$\begin{aligned} merge(C_1, C_2) &= \{I_1 \cup I_2, v_3\} \\ where, v_3 &= ((v_1 * |I_2|) + (v_2 * |I_2|))/(|I_1| + |I_2|) \end{aligned}$$

4 Using PD-Querysets

To find the PD-Queryset of a user query q_u, we first find a canonical query that is a good surrogate for q_u. This is accomplished by mapping q_u and all the canonical queries to semantic space vectors by using the unweighted semantic mapping equation, $\vec{q} = q^T U_r$. In the semantic space, we pick the canonical query vector that has the maximum *cosine similarity* to $\vec{q_u}$. Cosine similarity is computed as $\frac{\vec{q_1} \bullet \vec{q_2}}{||\vec{q_1}||\ ||\vec{q_2}||}$. The canonical queries, as well as cover queries from the PD-Queryset(Q_p) containing $\vec{q_u}$, are then issued to the server in a random order. The client receives the results from the server and filters the results from the canonical query corresponding to $\vec{q_u}$ using the original user query q_u (e.g., using tfidf). In the setting of a web search, the filtering could be done on the webpage summary text to re-rank the links. The steps of a PD-Search are listed in Algorithm 3. To achieve sufficient recall, we must retrieve more documents using $\vec{q_u}$ than with the (presumably more precise) q_u. This load is computational, imposed on the server and client; the final filtering step "hides" this from the user.

Algorithm 3 Steps in Plausibly Deniable Search

Require: The original query q_u of user

1: Map q_u to the semantic space vector $\vec{q_u}$
2: Find the canonical query that has the maximum Cosine Similarity score with $\vec{q_u}$; get its corresponding PD-Queryset Q_p containing cover queries
3: Submit Q_p to the search server to get the results $R(Q_p)$
4: Use q_u to filter $R(Q_p)$ to get the relevant documents for the original user query

If none of the terms from user query are in the terms list (pre-defined vocabulary) used in constructing the PD-Querysets, then the semantic mapping is not possible. However, if the vocabulary is sufficiently large, then this is unlikely. Moreover, if a term is not in a (sufficiently large) pre-defined vocabulary, it is likely to be very specific to the user intent and would make the query stand out; in this case the user can be warned of the potential privacy risk posed by issuing the query.

4.1 Arguing for Deniability. It is straightforward to show that the formal part of the second criteria (deniability) of Definition 2.4 holds. Given a PD-Queryset, any user query that maps to any canonical query in that set will produce the same PD-Queryset. While there may be multiple such queries, the seed queries clearly form such a set. While this does not quite ensure equal probability (for example, there may be many possible queries that map to one canonical query, while only a few map to the other), such probabilities are probably less critical than the plausibility argument (and may be intractable to compute; this is an area for

future work); we argue that the *existence* of k diverse queries mapping to the same PD-Queryset is sufficient.

Diversity of topics can be formally measured using Definition 2.2. We report an experiment on diversity of cover queries using pre-classified webpages in Section 5.4. Plausibility is more of a challenge: How do we evaluate if a canonical or cover query is more plausible? One possibility is experimentation with a large set of query logs; however defining and reporting such a novel experiment is beyond the space constraints of this paper and is left as future work. Perhaps the most immediate issue is coverage / recall. If this approach is unable to retrieve the information desired by the user, it will be of little value. We now give an evaluation of the retrieval characteristics of this approach.

5 Experimental Analysis

We used the DMOZ [10] categorized weblinks in the Resource Description Framework (RDF) format as the dataset for our experiments. The DMOZ Open Directory Project has a human-edited directory for all the classified web links, constructed and maintained by a community of volunteer editors. It uses a hierarchical category system to assign a web-link to a category, such as "Top/Computers/Algorithms", "Top/Sports/Football/American", or "Top/Science/Biology/". Since Top appears in any category, we ignore it in our analysis. The length -$|t|$ of a category t is defined as the number of sub-categories it contains. For example, |Computers/Algorithms| is 2 and |Sports/Football/American| is 3.

We selected and retrieved the websites from the DMOZ categories Computers, Science and Sports. This resulted in 314,130 web pages and 1,273,101 unique terms after stop word removal. The Lemur toolkit[17] was used for indexing and retrieval purposes. The term-document matrix was decomposed using a SVD tool [4] to get the U_r (i.e., term to semantic space) matrix keeping 30 dimensions as explained in Section 3.1. Table 1 lists the details of the document collection. There were 115,347 web pages under the category "Computers", 99,930 under "Science", and 98,853 under "Sports". The average number of unique terms in each document in the collection is 360. All the experiments were performed on a Linux machine running on 1.86GHz Intel Xeon E5320 quad core processor.

5.1 Canonical Query Generation. We ran the apriori frequent pattern mining [5] algorithm on the term-document matrix with the support $\Delta = 500$ and the stopping level as 2. This resulted in 2,659,249 seed queries: 8,651 single terms and 2,650,598 term pairs. These seed queries were converted to points in a 30 dimension semantic space. As described in Algorithm 1, these points were indexed through a KDtree data structure using the Weka [24] software and the nearest neighbor searches were performed (Algorithm 1). It took approximately 50 CPU hours to complete this step. This produced 931,863 unique canonical queries. Table 2 shows counts for various lengths of canonical queries. The majority of the canonical queries have 3 or 4 terms; the average length is 3.7. In [15], it is reported that the average length of web search queries is 2.4. The difference is mainly due to the number of real queries with one or two terms; we found single term queries did not make good canonical queries. As reported in a query log study [15], 92.2% of queries contain 1-4 terms; 91% of the generated canonical queries contain 1-4 terms. Thus the canonical queries are structurally comparable to actual web queries. In Section 5.3, we analyze the coverage of these canonical queries.

Table 1: Data Description

Total documents	314,130
Total unique terms	1,273,101
Number of documents in Computers	115,347
Number of documents in Science	99,930
Number of documents in Sports	98,853
Average document length	360

Table 2: Canonical Queries Generation (Step 1)

Total Seed queries	2,659,249
Single term seeds	8,651
Term Pairs seeds	2,650,598
Total Canonical Queries	931,863
Canonical queries(2 terms)	1019
Canonical queries(3 terms)	324,034
Canonical queries(4 terms)	524,256
Canonical queries(5 terms)	67,664
Canonical queries(6 terms)	14,890
Average terms per query	3.7

5.2 PD-Queryset Construction. The ACP Algorithm was then run against the canonical queries. For the (dis)similarity equations 3.3 and 3.4, we first computed the constants α, β and γ (shown in Table 3). For the neighborhood count, we used a web search query log containing 876,498 web search queries. These queries were mapped to points in the semantic space (Equation 3.2) and the neighborhood count was calculated for each canonical query. The difference vector $\vec{\delta}$ (Section 3.3.1) constructed by assigning $.005 * length(dim_i)$ to the ith component, where $length(dim_i)$ is the difference between the max and min values in the i-th dimension in Q_C. The ACP algorithm was run for up to two levels

Table 3: PD-Queryset Construction (Step 2)

α	4.8E-4
β	2.9E-4
γ	176,317
Number of Level 1(k=2) PD-Querysets	465,931
Number of Level 2(k=3,4) PD-Querysets	232,965

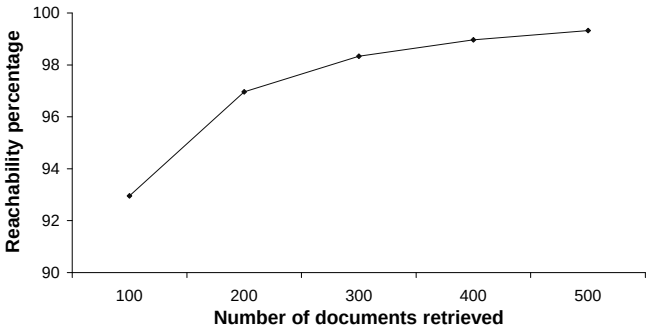

Figure 1: Reachability vs. numbers of documents retrieved per query

to create PD-Querysets with 2 and 4 canonical queries. The computations took about 350 CPU hours.

5.3 Coverage Analysis. The first test of coverage is to see whether all the documents are *reachable* through canonical queries. Reachability improves as the number of documents retrieved for a query increases; this is a tradeoff between recall and load on the server (to process the query) and client (to filter the results.) A document is reachable through a canonical query if the document is in the first N ranked documents in the retrieved set. Figure 1 shows the percentage of documents reachable through at least one of the canonical queries for various numbers of documents retrieved by the canonical query. With each query retrieving 100 documents, 93% of documents from the collection are reachable. The reachability percentage increases to 99.4% retrieving 500 documents/query. This shows that there exists a canonical query for retrieving almost every document in the collection.

As shown in Figure 2, the remaining 7% of the documents that were not retrieved in 100 documents/query had average document length of only 32. This is very small compared to the average document length of whole collection, 360. Figure 2 shows the average document lengths of unreachable documents for various numbers of documents retrieved.

The second aspect of coverage is recall: Do the canonical queries retrieve the documents that would have been obtained by the actual user queries? We selected 5000 random queries from the AllTheWeb

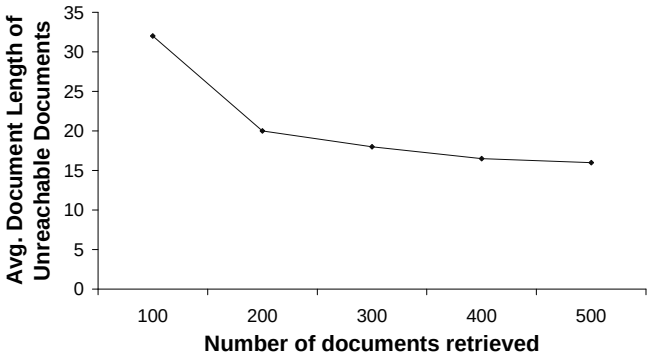

Figure 2: Average document length for unreachable documents (Collection average=360)

2001 [15] query log, a set of web-queries collected over 4 months through the alltheweb.com search engine. (Note that a completely different query log, from a different source, was used to generate the neighborhood counts in Section 5.2.) After the preprocessing stage, only the queries that contained at least 75% of terms from our collection's vocabulary were kept. This resulted in 3353 unique queries. (The 18% of queries that did not contain terms in the collection vocabulary would be deemed "risky" from a deniability viewpoint.) Using Algorithm 3, we obtained the closest canonical query to each of these queries. For retrieval, we used the tfidf-based retrieval scheme of Lemur toolkit. [3]

Since the query-document relevancy file is not available for these queries, we use the top 20 documents retrieved by the original queries as the relevant documents. We measure the recall of canonical queries by finding the intersection size of the retrieved documents between the original queries and canonical queries. Figure 3 shows the percentage of canonical queries that contained a given range of documents from the top 20 documents retrieved by the corresponding original queries, for a range of 100 to 500 documents retrieved by the canonical queries. Over half the time, the top 100 retrieved by the canonical query include the top document retrieved by the original query, and at least six of the top 20. Retrieving 500 documents gives on average over 11 of the top 20, and 69% of the queries contain at least 6 of the top 20. The percentage of canonical queries with poor recall (five or fewer of the top 20) rapidly declines as the number of documents retrieved increases, and the number with high recall goes up. Thus the canonical queries, though are not exactly same as the user queries (providing privacy / deniability), retrieve

[3] While we could have used an LSI-based retrieval method, this would give unfairly optimistic numbers. Using tfidf for retrieval/relevancy measurement is a better surrogate for real-world use, where the search engine technology is separate from that used to generate the matrix for generating/selecting canonical queries.

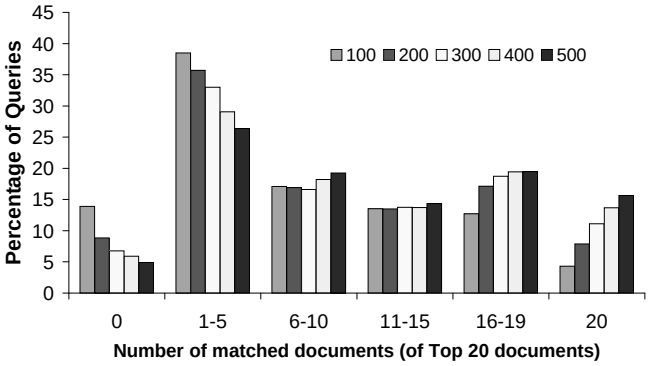

Figure 3: Percentage of queries for various number of common documents retrieved by the canonical queries and user queries

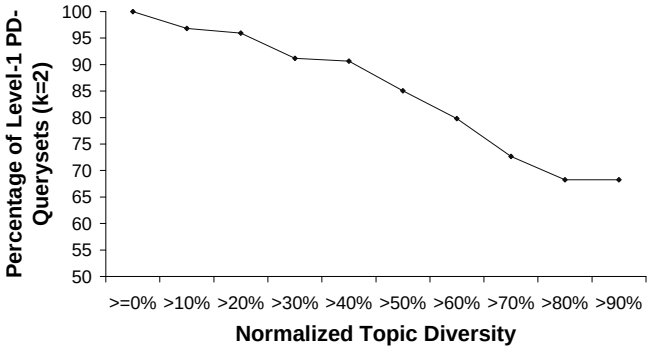

Figure 4: Percentage of Level-1 PD-Querysets for various Topic Dissimilarity

relevant documents at some (computational) expense of retrieving additional documents for filtering by client.

5.4 Plausible Deniability Analysis. For plausible deniability to be of value, the cover queries must be reasonably unrelated. In this analysis, we measure the diversity of the PD-Querysets. While we introduced Definition 2.2 to measure this, the hierarchical nature of DMOZ categories makes this a poor choice. For example, queries retrieving documents from sibling categories low in the hierarchy are not really diverse, but would show as diverse under Definition 2.2.

Instead, we use a measure that takes into account overlap in the hierarchy to determine query diversity. We first take the DMOZ categories of the top 10 documents in the search results of each query. From this list, we pick the longest prefix that occurs in more than 30% of the categories as the category of a given canonical query. Using this, we next measured the diversity of level-1 PD-Querysets. Let t be a category and $|t|$ is the length function. We define the *common path (cpath)* between two categories as the common prefix between them. For example, the common path between the categories "Sports/Football/American" and "Sports/Motorsports" is "Sports". The distance ($dist$) between two categories is $t_1| + |t_2| - 2 * |cpath(t_1, t_2)|$, the number of the number of nodes traversed to reach from t_1 to t_2. The normalized topic dissimilarity is computed as $\frac{dist(t_1,t_2)}{|t_1|+|t_2|}$. Figure 4 shows the percentage of level-1 PD-Querysets with $> p\%$ topic dissimilarity. As shown in the figure, 85% of the PD-Querysets contain canonical queries that have topic dissimilarity $> 50\%$. This shows the effectiveness of the ACP algorithm in producing topically diverse PD-Querysets.

We now show a set of queries produced during this experiment. The alltheweb search query {synthesis technology} mapped to the canonical query {synthesis molecular technology}. This query retrieved 15 of the top 20 original query documents in the first 300. The cover query was {baker priority report}. Note that both appear plausible, and are on the completely unrelated topics of technology and world politics.

5.5 Pattern Analysis on PD-Querysets. A sequence of queries on a particular topic would stand out as likely to be from a real user. While a formal definition of plausible deniability of a query sequence and designing effective solutions is left for future work, we show a sample analysis demonstrating that our approach is resistant to such attack. We start with a randomly selected query log containing 264 queries, ordered by timestamp. Applying the PDS system with $k = 4$ produces 1056 queries in 264 PD-Querysets. We devised a k-means clustering algorithm to group queries in to clusters by using the normalized topic dissimilarity (Section 5.4) as the distance function. A cluster with just (or a majority of) user queries would show a lack of diversity in cover queries; this would be mitigated if sufficient sequences of cover queries show a similar clustering. Figure 5 shows the number of user queries and cover queries in each cluster. The mean normalized topic similarity of queries to the cluster center in these clusters was 80%, which indicates the queries are tightly clustered It can be noted from Figure 5 that the clusters contain a mix of user queries and cover queries.

A sequence of queries is a collection of 2 or more queries such that they are not separated by more than 1 query in the log ordered by the timestamp. In each of these clusters, are there sequences of queries that may lead to identification? To answer this question, we compute the number of user query sequences and cover query sequences in each cluster (Table 4). While cluster 7 has a high percentage of user queries (24 out 40) (Figure 5), it has 2 user query sequences and also 2 cover query sequences. Notice that the clusters 1,2,3 and 5 have a high number of cover query sequences compared

Table 4: Number of Query Sequences in clusters

Cluster Number	1	2	3	4	5	6	7	8	9	10
User	2	3	13	1	3	0	2	0	1	0
Cover	13	12	33	1	21	2	2	3	0	11

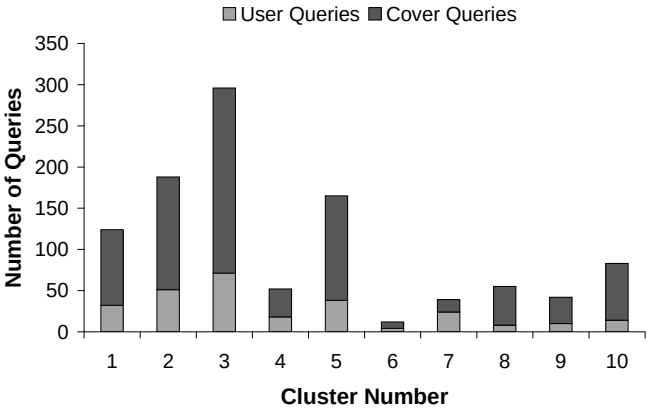

Figure 5: Queries in each cluster for k=4

to the user query sequences. While further study needs to be done with regards to sequential queries, these results show the promise in the proposed system.

6 Background and Related Work

The perfectly private solution to this problem is through *Private Information Retrieval*[8] (PIR). Unfortunately, the high complexity of PIR techniques, and the inability of search engines to do targeted advertising, prevent its practical adoption for search engines. The alternative of deniability is not unfamiliar to computer science literature, appearing in Deniable Encryption [6] and Deniable Authentication [20], key exchange protocols [18], and steganography [7]. We bring the basic concept to the real-world problem of web search. While our approach does impose extra cost on the server, it is linear in the amount of deniability required, and does not significantly impact the effectiveness of the marketing (assuming a "pay for click-through" model, although providing plausible deniability for links clicked remains a problem for future work.) Perhaps more important, it is purely a client-side solution; cooperation on the part of the server is not required.

6.1 Server-controlled Privacy. Users largely assume that servers will protect their privacy. After the AOL incident [2], several schemes were proposed for query log anonymization. These propose to make re-identification difficult by anonymizing identifiers such as IP address and userID. Unfortunately, the query text in the anonymized logs can still identify the individual who issued the queries [2]. It has also been shown that very accurate classifiers can be built for query logs to identify the gender, age, and location of users [16]; even with the removal of explicit identifiers, the classifiers have high prediction accuracy.

In [1], two solutions were proposed for anonymizing a query log. The first solution uses a *secret sharing* primitive to split a query into t parts, and enforcing the constraint that at least t users should have issued the query in order for it to be known. In the second solution, the actual user id is replaced with a different random id for each set of queries on different topics, thus not revealing the whole variety of interests of a particular user, and making re-identification more difficult. Search using peer-to-peer networks and anonymous surfing [23] can be used to break the link between the server and the user. However, these solutions do not address re-identification through the semantics of the queries.

6.2 User-Controlled Privacy. User-controlled privacy has received comparatively little attention. In [22], the authors categorize privacy protection in personalized search into four levels. While there has been work on hiding the query metadata from the server (e.g., Private Web Search [21]), little has been done to handle privacy problems inherent in query text. We divide these into two basic approaches: Query transformation, and the use of cover queries.

6.2.1 Query Transformation. In this approach the user applies a transformation $Q' = T(Q)$, and submits Q'. After getting the results, the user filters out the relevant documents for Q. Unfortunately, this has little ability to hide the semantic meaning of the search query, as $T(Q)$ must return similar results to Q to be effective. While the exact original query may be hidden, the *user intent* can still be identified. The PDS solution in this paper is also a form of query transformation, but with a clear statement of the privacy provided.

6.2.2 Random Queries for Query Log Obfuscation. Inserting traffic noise to cover real network traffic has a substantial history, the Trackmenot system attempts to do the same with queries [14]. A random query generator sends out queries from a user's machine to the search engine so that random queries act as noise and prevent queries being linked to the user. As the authors of this tool admit, this scheme does not guarantee privacy. While such an approach may be effective at masking encrypted traffic, a human can easily distinguish random queries from actual user queries. This

approach also imposes a high load on the server.

7 Conclusions and Future Work

We have shown a novel approach to protect a user's privacy when using search engines. This is very much a preliminary approach; in addition to the need to develop comprehensive evaluation methods for cover query plausibility there is substantial room for improved retrieval performance. Open challenges include:

Definitions: The current definitions are based on information-theoretic protection against attack; in practice, protection against a computationally bounded adversary should be sufficient. Definitions that appropriately capture computational reversibility appear to be non-trivial. However, such a definition could enable more flexibility than the canonical query approach.

Preprocessing: The current approach requires precomputing canonical and cover queries. Computing these on-the-fly, while still preserving the deniability arguments, is a challenging open problem. This would also allow for a greater variety of canonical queries, which could improve recall and enable fewer documents to be retrieved, lowering the load on the server.

Sequential Queries: While we have shown that identifying query sequences from plausibly deniable search sequences is not trivial, a formal definition and methods to provide deniability across a sequences is still an open challenge. While a system like PWS [21] provides some help (breaking metadata links between queries in a sequence), further exploration is needed.

Protecting privacy at the client gives stronger protections than current approaches. It also protects the search engines, as the information they receive would not be as sensitive as what they hold today - information from multiple sources (e.g., ISPs and external information on user interests as well as search engines) is needed to reconstruct the level of private knowledge now exposed by query logs. Putting control over user's privacy into the hands of the user will increase the freedom of users to use web search as they wish.

References

[1] E. Adar. User 4xxxxx9: Anonymizing query logs. In *Query Log Workshop, WWW*, 2007.

[2] M. Barbaro and J. Tom Zeller. A face is exposed for aol searcher no. 4417749. *The New York Times*, Aug. 9 2006.

[3] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against tor. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 11–20, NY, USA, 2007.

[4] M. Berry, T. Do, G. O'Brien, V. Krishna, S. Varadhan, and D. Rohde. SVDLIBC: C library for SVD.

[5] F. Bodon. A fast apriori implementation. In *IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, Melbourne, Florida,USA, 2003.

[6] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. *Lecture Notes in Computer Science*, 1294:90–104, 1997.

[7] M. Chapman and G. I. Davida. Plausible deniability using automated linguistic stegonagraphy. In *Proceedings of the International Conference on Infrastructure Security*, pages 276–287, London, UK, 2002.

[8] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[9] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[10] DMOZ: Open directory project.

[11] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *Commun. ACM*, 42(2):39–41, Feb. 1999.

[12] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, 1989.

[13] K. Hafner and M. Richtel. Google resists U.S. subpoena of search data. *The New York Times*, Jan. 20 2006.

[14] D. C. Howe and H. Nissenbaum. Trackmenot

[15] B. J. Jansen and A. Spink. An analysis of web searching by european alltheweb.com users. *Inf. Process. Manage.*, 41(2):361–381, 2005.

[16] R. Jones, R. Kumar, B. Pang, and A. Tomkins. "I know what you did last summer": query logs and user privacy. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 909–914, NY, USA, 2007.

[17] The lemur toolkit for language modeling and information retrieval.

[18] W. Mao and K. G. Paterson. On the plausible deniability feature of internet protocols. Technical report, Royal Holloway University of London, 2002.

[19] M. Murugesan and C. Clifton. Plausibly deniable search. In *Workshop on Secure Knowledge Management (SKM 2008)*, 2008.

[20] M. D. Raimondo, R. Gennaro, and R. Gennaro. New approaches for deniable authentication. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 112–121, NY, USA, 2005.

[21] F. Saint-Jean, A. Johnson, D. Boneh, and J. Feigenbaum. Private web search. In *Proceedings of the 6th Workshop on Privacy in the Electronic Society*, pages 84–90, Alexandria, Virginia, Oct. 29 2007. ACM Press.

[22] X. Shen, B. Tan, and C. Zhai. Privacy protection in personalized search. *SIGIR Forum*, 41(1):4–17, 2007.

[23] Tor: anonymity online.

[24] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, Oct. 1999.