

Plausibly Deniable Search *

Mummoorthy Murugesan
Department of Computer Science
Purdue University, W. Lafayette
mmuruges@cs.purdue.edu

Chris Clifton
Department of Computer Science
Purdue University, W. Lafayette
clifton@cs.purdue.edu

Abstract

Query-based web search is becoming 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 could be identified from the queries alone [6]. Government requests for such logs serves to increase the concern. We propose a client-centered approach based on plausibly deniable search: actual user queries are replaced with a set of queries that hide the actual query. By using a singular-value decomposition approach (demonstrated on TREC-4), we are able to generate cover queries that have characteristics similar to the actual user query (although on unrelated topics), preventing the actual query from "standing out" from the cover queries.

1 Introduction

Search engines such as Google, Yahoo!, and MSN boast huge user bases. Logs of the queries can give extensive insight into people's interests and activities. This data can be used in the aggregate, but can also be used to develop profiles of individuals, raising concerns about the privacy of users.

Anonymizing the logs does not solve this issue. In July 2006, AOL released an anonymized search query log [6] 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, but assigning random ID for each user. However, this simple anonymization proved ineffective; the *query itself* often

*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.

contained identifying information [6] (e.g., ego-surfing).

Adding to this concern is recent government attempts to obtain query logs. The U.S. Government has subpoenaed search logs from the major search engines [3]. While the subpoena did not request identifying information, it 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; witness the use of information obtained from Yahoo! in the jailing of a Chinese dissident, and the aftermath [13]. We are aware of only one that addresses protecting search text [9]. We discuss this and other related work in Section 4.

1.1 Plausibly Deniable Search

We instead address this by having the client generate a set of cover queries for each real query. This eliminates timing-based attacks to discover the real query (a problem with [9]). While this does not completely hide the real query, if done properly it supports a notion of *plausible deniability*. While it is possible than an individual may have issued the actual query, it is equally possible that they issued one of the generated cover queries. This places a burden of proof on the holder of the query log when trying to use that log to imply something about an individual's interests.

More formally, assume a server log has a set of queries $S = \{Q_1, \dots, Q_k\}$ from a particular user at a particular timestamp. A party with access to this log identifies the user, and tries to prove that the user searched for 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:

Definition 1 (*PD-Privacy*) A user 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 the same probability as Q_i ,
2. all $Q_j \in S$ are on different topics, and
3. all $Q_j \in S$ are equally plausible as a real query.

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 come from a predefined list, while allowing arbitrary user queries, would not meet this test.

For example, assume that an user issues a query ‘galaxy sun’, and this scheme generates a masking query ‘house garage’. Both queries are plausible, and they seem to be completely unrelated. If these two 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 *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.

The approach taken by this paper is to use LSI based semantic mapping to generate a fixed set of possible queries, avoiding the problem of identifying the real query as the one that wouldn’t be generated by the method, and fixed sets of cover queries to meet constraint 1. In Section 2 we detail the approach, followed with an empirical demonstration on TREC-4 queries in Section 3. We discuss related work and other approaches in Section 4.

2 Construction of PD-Querysets

We now present a method for statically constructing PD-Querysets based on a set of seed documents, which are representatives of the information being retrieved. The idea is that we extract terms from these 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) 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.

2.1 Latent Semantic Indexing

The use of SVD to map terms in a topic space forms the basis of Latent Semantic Indexing (LSI)[8]. LSI ad-

resses 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 be A of size $n \times m$ 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 of a matrix A gives us three matrices such that

$$A = USV^T, \quad (1)$$

where U is an $n \times n$ left singular matrix, S is an $n \times m$ diagonal matrix, and V is an $m \times m$ right singular matrix. 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 $r \times m$ V_r^T are generated. 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:

$$\vec{q} = q^T U_r S_r^{-1}, \quad (2)$$

where \vec{q} is the vector representation of q in the concept space. User queries are mapped to canonical queries by finding the nearest neighbor in this semantic space.

2.2 Seed Terms and Canonical Queries

To ensure a reversible mapping of queries to cover queries (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 $|T|$. Instead, we use a two-step process to generate canonical (and cover) queries.

We start with seed queries consisting of terms, pairs of terms, and larger “likely” groups (terms with high tfidf scores for each document.) These seeds are mapped into the semantic space. However, these term pairs are insufficient to represent actual user intent, and the larger tfidf based sets could stand out, we generate a canonical query (and likewise a cover query) corresponding to each seed query as follows:

1. Merge seed query points that are within a specified Euclidean distance. This merge is accepted only if the total number of terms from the queries of the points do not exceed a threshold value (set to value between 5 and 30)
2. Continue Step 1 until no more points can be merged.

3. In each group of points merged together, create the canonical query as the union of seed query terms.

Given a set of seed queries, the above steps produce canonical queries. For any user query, the corresponding canonical query closest in the semantic space is selected. The canonical query contains sufficient specificity to act as a reasonable surrogate for the actual user query.

2.3 Distance Function

To form a PD-Queryset, we need to combine canonical queries 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 . We use the relative density of queries from 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.)

We now define a distance function between canonical as a sum of following three components:

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

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

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 given by,

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

3. Neighborhood Count: The neighborhood 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} .

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

The neighborhood count function $nhc()$ takes \vec{q} , a query log Q_L , and $\vec{\delta}$, and returns the number of queries from Q_L that fall inside the hypercube. $\vec{q}[i] \pm \vec{\delta}[i]$.

The distance between two query vectors is defined as follows:

$$dist(\vec{q}_1, \vec{q}_2) = (1 - edist(\vec{q}_1, \vec{q}_2)/\alpha) + \frac{\|\vec{q}_1\| - \|\vec{q}_2\|}{\beta} + \frac{|nhc(\vec{q}_1) - nhc(\vec{q}_2)|}{\gamma} \quad (3)$$

where,

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

Thus, queries with similar neighborhood, similar magnitude, but far apart in the semantic space get a lower distance score and would be placed in the same PD-Queryset.

Similarly, we define the distance metric between a set of queries. Given that there are two sets of queries, $A = \{\vec{a}_1, \dots, \vec{a}_n\}$, $B = \{\vec{b}_1, \dots, \vec{b}_m\}$ the distance between them is calculated as follows: ($|A| \geq 1$ and $|B| \geq 1$)

$$dist(A, B) = (1 - \alpha_1/\alpha) + \beta_1/\beta + \gamma_1/\gamma \quad (4)$$

$$where, \alpha_1 = MIN(edist(\vec{a}_i, \vec{b}_j)), \forall i, j$$

$$\beta_1 = |(\sum_{i=1}^n \|\vec{a}_i\|)/n - (\sum_{j=1}^m \|\vec{b}_j\|)/m|$$

$$\gamma_1 = |(\sum_{i=1}^n nhc(\vec{a}_i))/n - (\sum_{j=1}^m nhc(\vec{b}_j))/m|$$

2.4 Agglomerative Construction of PD-Querysets (ACP)

We present a 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 1, we first construct a Level-1 multiset of pairs of vectors. We compute a distance matrix between all pairs of queries in Q_C . From the distance matrix, we select the query pairs in ascending order of distances and add them to the set L_1 . For some query pair (q_i, q_j) with the distance d_1 , either q_i or q_j could have been part of a better match with distance $d < d_1$. If so, then q_i or q_j must be already in L_1 , and so the next pair with the lowest distance is selected. Having constructed L_1 , the next level sets (L_2, L_3 , etc.,) are produced by merging the sets from one level below. We also ensure that the resultant sets contain non-overlapping cluster of queries. Given that we need at least k queries in the resultant cluster, we have the condition that $2^l \geq k$, giving us the stopping criteria for the merging procedure as $l < \log_2 k$.

2.5 Using PD-Querysets

To find the PD-Queryset of a user query q_u , it is first mapped to a semantic space vector \vec{q}_u . In the semantic

Algorithm 1 The ACP Algorithm

Require: $Q_C = (\vec{q}_1, \dots, \vec{q}_n), k$; k is the privacy parameter; Query log Q_L .

- 1: Compute the $n \times n$ distance matrix using the function $dist(\vec{q}_1, \vec{q}_2)$;
- 2: $L_1 \leftarrow \emptyset$
- 3: **while** $|L_1| < n/2$ **do**
- 4: Get query pair (q_i, q_j) with the next lowest distance
- 5: **if** q_i or q_j is not already added to L_1 **then**
- 6: Create a cluster $C = \{q_i, q_j\}$, 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 $dist(C_i, C_j)$ 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: $l \leftarrow l + 1$; $\{L_l$ contains the final PD-Querysets}
- 18: **end while**

space, the closest canonical query \vec{q}_u' is selected, including the PD-Queryset containing \vec{q}_u' . The corresponding canonical queries / cover queries are then issued to the server in 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). The steps of a PD-Search are listed in Algorithm 2. To achieve sufficient recall, we need to retrieve more

Algorithm 2 Steps in Plausibly Deniable Search

Require: The original query q_u of user

- 1: Map q_u to semantic space vector \vec{q}_u
- 2: Find the closest canonical query that has the minimum Euclidean distance from \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

results using \vec{q}_u' than would be required 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. In the next section, we discuss this in detail by constructing canonical queries for TREC-4 queries, and analyzing the results.

3 Demonstration

We now present an evaluation using the TREC-4[1] dataset, consisting of 557,674 news articles, and 567,421 unique terms. Queries are mainly topic descriptions, consisting of 10 terms in average. Lemur toolkit[2] is used for indexing and retrieval(tfidf) purposes. The seed queries for the results presented here consist of: 1) 567, 421 individual terms, 2) The top 2, top 5, and top 10 terms for each document (as ranked by tfidf scores), and 3) a random subset of all possible pairs. This gives approximately 2.5 million seed queries, and 31217 canonical queries (each with more than 10 terms), a practical limit with the tools we are using.

This turned out to be a significant limiting factor: the approached worked reasonably well for about 20% of the TREC-4 queries, but returned NO relevant results for the others. This is because the incomplete seed query set limited the coverage of the semantic space, and thus many queries do not have a good corresponding canonical query.

We present results only for the queries for which the canonical query retrieved some documents; we believe that once future work overcomes system limits on the number of canonical queries this will provide a reasonable lower bound on what could be achieved.

3.1 Coverage

Standard precision/recall curves are not an appropriate measure given the two-stage (server/client) filtering process; if we retrieved enough documents from the server, the actual user view of the data would be unchanged. Instead, we want to measure the increased load required to get reasonable results: How many documents do we need to retrieve from the server using the canonical query to get comparable results to the actual query? To show this, we instead report precision for varying “user view” result sizes for various “expansion factors” (number of documents retrieved from the server using the canonical query.) This provides a reasonable surrogate for the actual concerns in our web search scenario. Figure 1 shows average precision for various numbers of documents retrieved. Canonical queries with expansion factor greater than 400 achieve precision values comparable with raw queries.

Figures 2 and 3 show results on a per-query basis when looking only at the top 10 results for canonical queries returning 400 and 800 results, respectively. For N=800 (Figure 3), the precision after 10 documents retrieved is close to that of raw queries. For Query 226, the canonical query performs badly compared to the raw query, but in several other queries (202, 209, 246) retrieval performance improves. In both the experiments, Query 233 has a very low precision (2%), and that is mainly due to the original topic description given in the TREC-4.

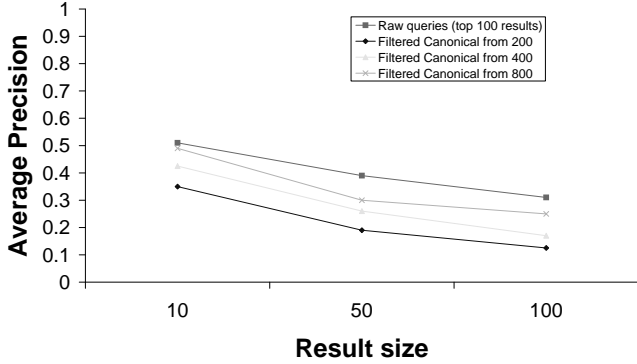


Figure 1. Precision at varying result sizes and expansion factors

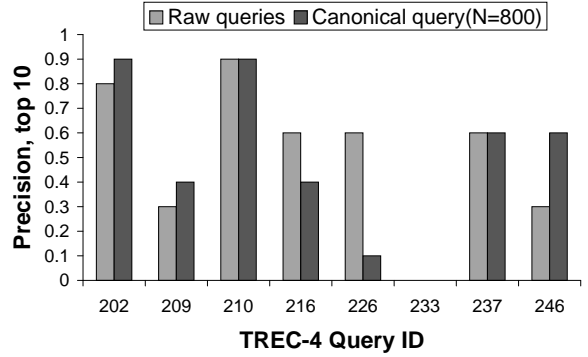


Figure 3. Comparison of Precision(at 10) of Raw queries Vs Canonical queries returning 800 items

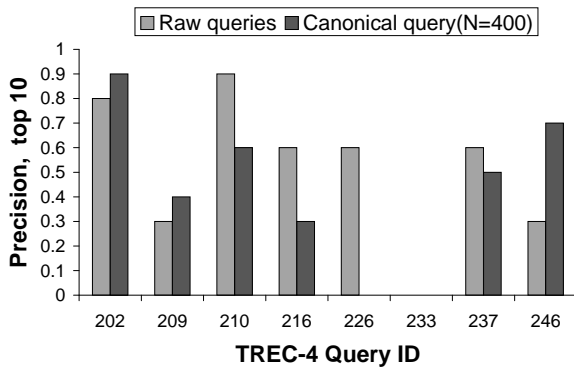


Figure 2. Comparison of Precision(at 10) of Raw queries Vs Canonical queries returning 400 items

3.2 Cover Queries

The TREC-4 dataset does not provide sufficient or appropriate data to fully evaluate plausibility and deniability of queries. Space limitations and the preliminary nature of this work prevent a full evaluation. Instead, we randomly selected two of the TREC-4 queries for which we provide coverage results, and present the 3-deniable canonical and cover queries in Table 1. It can be observed that cover queries of Q-226 are from two completely different topics: cover-1 query is about computers (db2, sun, LAN, sever), and cover-2 query is about baseball (molitor, eckersley). Thus, this PD-Queryset satisfies our requirements for being issued to the server, and gives us the plausibly deniable search property.

4 Background and Related Work

The perfectly private solution to this problem is through *Private Information Retrieval*[7] (PIR). Unfortunately, for a

single server approach (and assuming the existence of a secure coprocessor) the best solutions require quadratic preprocessing and linear storage *per query*[5], and the business model of search engine companies that rely heavily on targeted advertising will get affected. 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.)

After the AOL query log release incident, there have been several schemes proposed for Query Log anonymization [4]. Peer-peer networks and anonymous[14] surfing could be used to break the link between the server and the user. However, it still does not address the re-identification from the query text problem. We view our approach as complementary to server-controlled privacy; giving plausible deniability against a curious server, and added protection when anonymized logs are released.

User-controlled privacy has received comparatively little attention. While there has been work on hiding the query metadata from the server (e.g., PWS[12]), little has been done to handle the privacy problems inherent in query text. The trackmenot system attempts hide the original user query by sending random queries at random intervals.[9]. While such an approach may be effective at masking encrypted traffic, a human can easily distinguish random queries from actual user queries.

5 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:

Table 1. Example canonical and cover queries

TREC-4	Actual query	Canonical	Cover 1	Cover 2
210	widespread illeg dispos medic wast done combat dump	wast dispos unincorpor garbag can ton crowther ocean parment dump	vincent sierra lacoss ammann cole higgin sixth mullin dreher	state salina unit trade mexico japan keiretsu relationship japanes estat billion
226	larg scale state allow lotteri gambli improv financi condit reduct note properti tax incom road	state worker oldest mcdonald 1987 incom chang declin deduct connecticut taxpay	LAN enhanc product protocol applic acit sun server interfac db2 cus-tomis	bailout eckerslei molitor brewer vaughn salazar tiger throw fisk

Preprocessing: The current approach requires pre-computing canonical and cover queries. Computing these on-the-fly, while still preserving the query equivalence/deniability arguments, is an open and challenging problem.

Retrieval Efficacy: Closing the gap between the query issued to the server and the actual user intent will serve to make this approach practical.

Sequential Queries: A sequence of queries on the same topic could lose the deniability argument; while each PDS-Queryset might be diverse, the preponderance of one topic across multiple PDS-Querysets would reveal user intent.

Protecting privacy at the client gives stronger protections than current approaches. It also protects the search engines, as the information they receive is not 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] "The fourth text retrieval conference (trec-4)." http://trec.nist.gov/pubs/trec4/t4_proceedings.html
- [2] "Lemur toolkit for language modeling and information retrieval: <http://www.lemurproject.org/>."
- [3] "Google resists u.s. subpoena of search data," *The New York times*, January 20, 2006
- [4] E. Adar, "User 4xxxxx9: Anonymizing query logs," in *Query Log Workshop, WWW*, 2007.
- [5] D. Asonov and J.-C. Freytag, "Almost optimal private information retrieval," in *Second International Workshop on Privacy Enhancing Technologies PET 2002*. San Francisco, CA, USA: Springer-Verlag, Apr. 14-15 2002, pp. 209–223.
- [6] M. Barbaro and T. Z. Jr., "A face is exposed for aol searcher no. 4417749," *The New York times*, Aug. 9 2006.
- [7] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM*, vol. 45, no. 6, pp. 965–981, 1998.
- [8] 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*, vol. 41, no. 6, 1990.
- [9] D. C. Howe and H. Nissenbaum, "Trackmenot: Browser extension." <http://mrl.nyu.edu/dhowe/trackmenot/>
- [10] R. Jones, R. Kumar, B. Pang, and A. Tomkins, "'I know what you did last summer': query logs and user privacy," in *CIKM '07*. New York, NY, USA.
- [11] R. Kumar, J. Novak, B. Pang, and A. Tomkins, "On anonymizing query logs via token-based hashing," in *WWW*, 2007.
- [12] 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*. Alexandria, Virginia, Oct. 29 2007, pp. 84–90.
- [13] The Associated Press, "Yahoo criticized in case of jailed dissident," *The New York Times*, Nov. 7 2007.
- [14] "Tor: anonymity online." <http://www.torproject.org/>