Quantifying Web-Search Privacy

Arthur Gervais[†], Reza Shokri[†], Adish Singla[†], Srdjan Capkun[†], and Vincent Lenders[‡]
[†]ETH Zurich, Switzerland, [‡]Armasuisse, Switzerland
[†]firstname.lastname@inf.ethz.ch, [‡]firstname.lastname@armasuisse.ch

ABSTRACT

Web search queries reveal extensive information about users' personal lives to the search engines and Internet eavesdroppers. Obfuscating search queries through adding dummy queries is a practical and user-centric protection mechanism to hide users' search intentions and interests. Despite few such obfuscation methods and tools, there is no generic quantitative methodology for evaluating users' web-search privacy. In this paper, we provide such a methodology. We formalize adversary's background knowledge and attacks, the users' privacy objectives, and the algorithms to evaluate effectiveness of query obfuscation mechanisms. We build upon machine-learning algorithms to learn the linkability between user queries. This encompasses the adversary's knowledge about the obfuscation mechanism and the users' web-search behavior. Then, we quantify privacy of users with respect to linkage attacks. Our generic attack can run against users for which the adversary does not have any background knowledge, as well as for the cases where some prior queries from the target users are already observed. We quantify privacy at the query level (the link between user's queries) and the semantic level (user's topics of interest). We design a generic tool that can be used for evaluating generic obfuscation mechanisms, and users with different web search behavior. To illustrate our approach in practice, we analyze and compare privacy of users for two example obfuscation mechanisms on a set of real web-search logs.

Categories and Subject Descriptors

[**Security and privacy**]: Privacy protections; E.0 [**Data**]: General

Keywords

Web Search; Privacy; Obfuscation; Quantification Framework; Query Privacy; Semantic Privacy; Machine Learning

CCS'14, November 3–7, 2014, Scottsdale, Arizona, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2957-6/14/11 ...$15.00.
http://dx.doi.org/10.1145/2660267.2660367 .

1. INTRODUCTION

Users search the web to obtain information or find websites. Through this, they leave a trail of their interests and intents. This information can be used by search engines and eavesdroppers to build a profile of users and to infer sensitive personal information about them [35, 22].

Privacy in web search can be protected in different ways. In a *system-centric* solution, we can design a search engine using private information retrieval, such that users can obtain the results to their searches without revealing their queries or their search activities to the search engine [20, 24, 11]. The benefit of this approach is that no information about users' search activities is revealed to the service provider or any eavesdropper. This solution, however, cannot protect privacy of users with respect to the existing popular search engines. In a *network-centric* solution, users can make use of anonymous communications to hide their identities with respect to the search engines, in order to make their queries unlinkable [31, 14]. This technique can prevent an adversary from constructing a profile for each user to some extent. Features extracted from the user's web browser can be used however to fingerprint the user and link her queries [16]. In a *user-centric* approach, users can conceal their real queries by issuing interleaving fake queries [1, 12, 21]. The challenge here is to generate fake queries that cannot be distinguished from real queries, as simple randomly generated queries can be easily filtered out from the set of observed queries from a user [9, 12, 28]. Note that there is a possibility of combining these approaches for designing a hybrid protection mechanism.

In this paper, we focus on evaluating user-centric web search query obfuscation methods. Despite the fact that a number of obfuscation mechanisms such as [1, 32, 26, 30, 37, 21] have been proposed so far, and there exists simple attacks to show their limitations [9, 12, 28], there is no common methodology and generic *quantitative* framework for measuring privacy of users for different obfuscation mechanisms. In this paper, we propose such a framework.

We construct a generic model for the users' web-search *behavior*. This determines the relation between queries of each user, in general. We model this using a similarity score function between query pairs that predicts whether any two queries could belong to the same user or not. We assume that adversary might have access to a dataset of real queries. We extract a variety of features from each query about its different dimensions including time, structure, content, and landing web pages. We then make use of gradient tree boosting regression algorithms to learn users' web-search behavior in the form of linkage functions [18, 19]. We also use

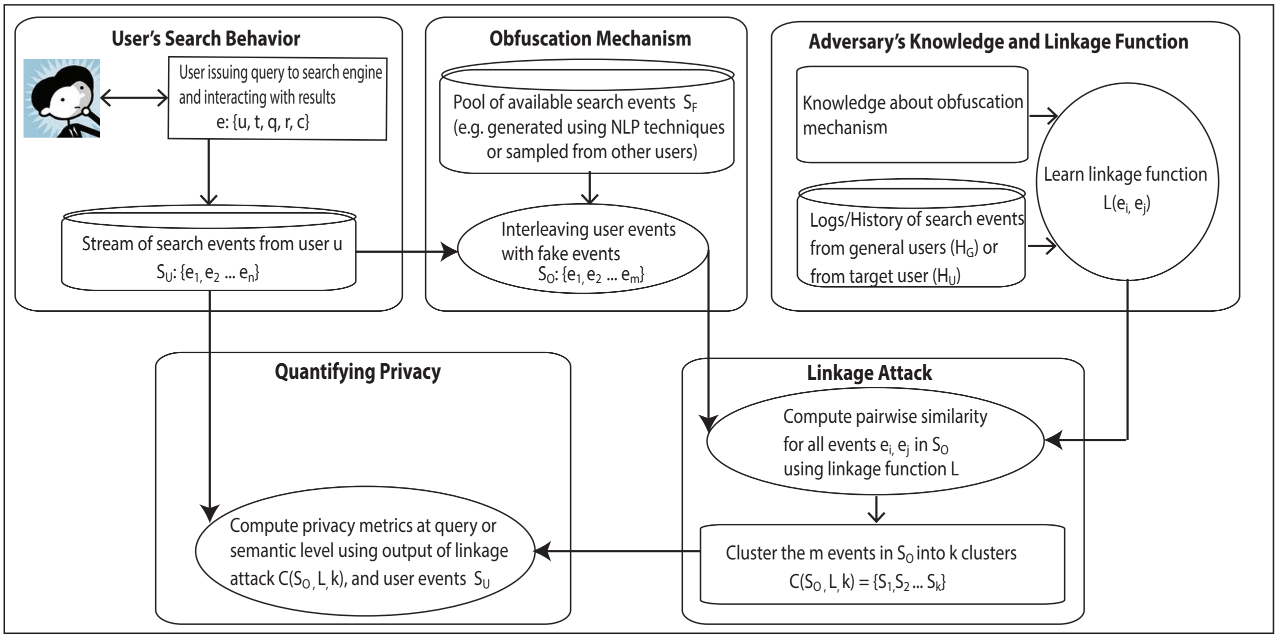
User's Search Behavior
User issuing query to search engine and interacting with results
e: {u, t, q, r, c}
Stream of search events from user u
S_U: {e_1, e_2 ... e_n}
Obfuscation Mechanism
Pool of available search events S_F
(e.g. generated using NLP techniques or sampled from other users)
Interleaving user events with fake events
S_O: {e_1, e_2 ... e_m}
Adversary's Knowledge and Linkage Function
Knowledge about obfuscation mechanism
Logs/History of search events from general users (H_G) or from target user (H_U)
Learn linkage function L(e_i, e_j)
Quantifying Privacy
Compute privacy metrics at query or semantic level using output of linkage attack C(S_O, L, k), and user events S_U
Linkage Attack
Compute pairwise similarity for all events e_i, e_j in S_O using linkage function L
Cluster the m events in S_O into k clusters
C(S_O, L, k) = {S_1, S_2 ... S_k}

Figure 1: Overview of our framework for quantifying web-search privacy.

our knowledge on the obfuscation mechanism in modeling this function. In addition to this generic model, depending on the evaluation's settings, we might also assume that the adversary has access to some history of the target user's web search behavior. This enables the adversary to construct a specific model for the target user.

We quantify web-search privacy of users against *linkage attacks*, where the attacker's goal is to find the link between queries issued by the target user and separate real queries from the fake ones. To this end, the attacker makes use of his user behavior and obfuscation model that are compressed in the form of linkage functions. In our framework, we rely on the results of the linkage attack to compute a user's privacy with respect to different objectives (i.e., privacy metrics). For example, privacy can be quantified at the query level and at the semantic level, depending on whether the user's objective is to hide the (linkability) structure of her queries or to conceal her topics of interest. The randomness of the user's behavior also contributes to the user's privacy against linkage attacks. Thus, to evaluate the privacy gain of using an obfuscation mechanism, we subtract the effect of user's randomness and compute the *relative* privacy of users.

We run our methodology on queries from the AOL dataset [27]. We consider two representative obfuscation mechanisms that either make use of bag of words or real user query logs to generate fake queries. We then evaluate and compare users's privacy with respect to different metrics. The results show that our attack can easily break privacy of majority of user, especially in inferring their topics of interest.

The main contributions of this paper are therefore:

- We propose a generic quantitative framework using which we can model attacks against web query obfuscation mechanisms as well as privacy metrics to capture the users' privacy objectives.
- We design the linkage functions that can model the web-search behavior of users in addition to that of the target user. The linkage function also captures the relation between fake and real queries, hence also models the obfuscation mechanism.
- We implement a linkage attack that splits the fake queries from the user's queries. By comparing the attack's result with the user's real set of queries, we quantify privacy in multiple dimensions, e.g., query trace structure and semantics.

The rest of the paper is organized as follows. In Section 2, we present the overall framework that we propose for quantifying web-search privacy. In Section 3, we present the dataset of web search queries that we are using throughout the paper. We also seek to understand the behavior of users with respect to their web search, and we extract features from their web queries that reflect the users' behavior. In Section 4, we build upon our user model and we present our methodology for quantifying privacy of users against linkage attacks. In Section 5, we use our quantification method to evaluate privacy of users (in our dataset) with respect to different obfuscation mechanisms. In Section 6, we survey the related work and put them in perspective with our contribution.

2. PRIVACY FRAMEWORK

In this section, we introduce our framework for quantifying user's web-search privacy. As shown in Figure 1, the framework is composed of the following main components: (i) user's search behavior, (ii) obfuscation mechanisms, (iii) adversary knowledge, (iv) linkage attack, and (v) privacy metrics. We now provide high level details of each of these components and introduce the required notation.

2.1 User's Search Behavior

Users issue queries to the web search engines to seek their information needs. In response, the search engine retrieves a result page consisting of a ranked list of web pages. The user interacts with the search result page by clicking and browsing the relevant documents, or by further refining the search query to fulfill the information needs. This interaction of the user with the search engine leaves a trace of her web search activity as a sequence of search query events. We model each such query event of a user along with its contextual information as $e : \langle u, t, q, r, c\rangle$, where u is the user identity (e.g., her username, IP address, cookie identifier, or any pseudonym that links the user's queries together but does not necessarily reveal her true identity), t is the time at which the query is issued, q is the user's query string which is composed of a sequence of terms, r is the search result page returned by the search engine containing ranked lists of web pages, and c is the set of pages that are clicked by the user so as to seek the required information. The web-search trace of the target user U, given by a series of web search query events from the user U, is denoted as $\mathcal{S}_U : \{e_1, e_2, \ldots, e_n\}$.

2.2 Obfuscation Mechanisms

User-centric obfuscation mechanisms aim to protect the privacy of a user by interleaving a set of fake query events with the real queries of the users. Let $\mathcal{S}_F$ be the set of query events associated with the fake queries that are used by the obfuscation mechanism for protecting the privacy of user U. An obfuscation mechanism might generate the fake trace $\mathcal{S}_F$ by observing the behavior of target user U, or the fake trace can be generated independently. The fake queries could also be generated by getting information from different sources. Another key parameter of the obfuscation mechanisms is the way interleaving of fake and real queries is done. For example, the obfuscation mechanism may send fake queries at regular intervals or send a burst of fake queries when a real query is issued by user. In Section 6, we provide a survey of the various existing obfuscation mechanisms and their characteristics.

In particular, we evaluate the following two types of obfuscation mechanisms in this paper:

- Mechanisms that generate fake queries by sampling from a bag of text. One example of such a mechanism is TrackMeNot (TMN) [1, 21] that mainly uses some RSS feeds for generating fake queries, and refines its queries by observing the search results of its own issued queries in the past. We choose TMN due to its popularity and open availability.

- Mechanisms that make use of real queries from a set of other users to generate fake queries for the target user. We consider a specific variant of such a mechanism that chooses one random user and uses all her queries for this purpose.

As a result of the obfuscation, the sequence of events coming from user U appears as $\mathcal{S}_O : \{e_1, e_2, \ldots, e_m\}$, obtained by interleaving $\mathcal{S}_U$ and $\mathcal{S}_F$. Hence, the search engine or any eavesdropper observes $\mathcal{S}_O$ from the user U, where all the events in $\mathcal{S}_O$ have the same identity U thus appearing as if they are issued by the target user.

2.3 Adversary's Knowledge

The goal of the adversary is to separate the fake queries from real ones, in order to extract accurate personal information about the user. We quantify the privacy of users by evaluating the effectiveness of the employed obfuscation mechanism against such attacks from the adversary. We assume the following about the prior knowledge of the attacker:

- **Obfuscation mechanism:** As a privacy evaluator, our objective is to assess the robustness of particular obfuscation mechanisms against strong attacks. Hence, we assume knowing the obfuscation mechanism or being able to observe its behavior prior performing the evaluation attack. Apart from knowing the exact mechanism, the adversary might additionally be aware of the parameters of how the fake queries are generated, and how the interleaving is done. Alternatively, if the adversary does not know such exact details about the obfuscation mechanism, we assume that he can infer the behavior of the mechanism by observing its output in an offline training phase.

- **Log history of users' search activities:** We further assume that the adversary has access to some log history of web search activities for a set of users (excluding target user), denoted by $\mathcal{H}_G$. From this dataset, the adversary can build a *generic model* for the users' web-search behavior and can learn the models needed for linkage attacks (as discussed further below).

- **Log history of the target user:** The adversary might additionally have access to some history of the target user U's query events, given by $\mathcal{H}_U$. This can additionally enable the adversary to build a more *specific model* for the target user, thus further empowering him to effectively predict the user's queries that she issues over time or the topics that she is interested in.

2.4 Linkage Function and Attack

The objective of the linkage attack is to partition the set of events in $\mathcal{S}_O$ and determine which of the query events are associated with the target user. By exploiting the adversary's knowledge about the obfuscation mechanism and the log histories, the key idea is to learn a linkage function $L(e_i, e_j)$ that quantifies the similarity of any two events e_i, e_j and uses it to determine whether they belong to the same target user or not.[1] The learning is done by extracting different contextual and semantic features from the available logs of the queries (and additionally from the fake auto-generated queries depending upon the type of obfuscation mechanism used) [36, 34]. These features are then used to learn (i) how query events of a generic user as well as the target user are correlated, and (ii) what feature value ranges represent the behavior of the target user. By using machine learning techniques to learn the linkage function, the overall framework easily adapts to new types of obfuscation mechanisms, to different data sets as well as different levels of prior knowledge of the adversary. The details of this learning procedure as well as the different features extracted are discussed in Section 3.2 and Section 4.

We use the linkage function L as the basis for the linkage attacks. Given a set of query events $\mathcal{S}_O$, the adversary first

[1] Note that in this paper we limit the study on tuples of two events, however more complex structures (e.g. cliques of three or more queries) might improve the attacker's accuracy. Nevertheless, the quantification methodology remains the same.

computes the pairwise similarity $L(e_i, e_j)$ between any two events $e_i, e_j \in \mathcal{S}_O$. These pairwise similarities then allow the adversary to cluster or partition the events $\mathcal{S}_O$ into a set of k clusters. Thus, the output of the linkage attack is a set of clusters, given by $C(S_O, L, k) = \{S_1, S_2, \ldots, S_k\}$.

In case the adversary has access to the target user's history $\mathcal{H}_U$, this information could further be used to label these clusters or learn more specific linkage functions. In this paper, we do not elaborate on this last step.

2.5 Privacy Metric

We quantify the privacy of users given the output of the linkage attack under various types of privacy sensitivities that the user might have. In general, we measure privacy in terms of the adversary's error in correctly constructing the user's profile.[2] This reflects the privacy risk of issuing web queries using a particular obfuscation mechanism.

2.5.1 Query Privacy

Let us consider that the user's objective is to hide the relation between her queries, so that the adversary could not infer the relation between her different interests (dimensions of her search profile). We quantify this by measuring the structural distance between the clusters generated by the linkage attack and the set of the user's real queries. Consider running the linkage attack by setting $k = 2$. A perfect partition by the adversary would result in $C(S_O, L, 2) = \{S_1, S_2\}$ where $S_1 = S_U$ and $S_2 = S_F$, thus completely separating the real queries of the target user from the fake queries introduced by the obfuscation mechanism.

In this case, we quantify privacy as to what extent the obfuscation function leads to the deformation of the structure of the queries when partitioned by the linkage attack. This is measured by computing how many pairs of query events from the target user end up in different clusters (i.e. the false negatives of the attack). Similarly, we measure the false positives of the attack by computing how many pairs of query events from the target user and fake events end up in the same cluster. These errors reflect different privacy gains of the user.

2.5.2 Semantic Privacy

As another objective, assume that the user wants to protect the privacy at a higher semantic level (instead of query level discussed above). Assume that we can map a set of query events to a semantic profile of interests (for example, quantifying that a user is a businessman or a technical person, captured through some histogram over a set of semantic topics). By constructing such a semantic profile for the sets S_U, S_1 and S_2, we measure the effectiveness of the obfuscation mechanism by quantifying the differences between the semantic profiles of S_U compared to that of the two clusters S_1 and S_2. The higher the distance is, the less semantic information about the user is leaked to the adversary. Similar approach could be applied for other higher level aspects of the user profile, for example, temporal aspects of the user search activity. The semantic privacy metric reflects how much information about the user's profile is in adversary's estimate, i.e., the mutual information between user's profile and the estimated profile. This is similar to the information-theoretic mutual information metric, with the difference that we do not need to estimate the probability distribution over the profiles (which is very difficult if not impossible [9]).

AOL Dataset	
Total number of users	657,426
Total number of queries	21,011,340

Our Sampled Dataset	
Total number of users	100
Total number of queries	73,984
Average number of queries per user	828.62 (±133.54)

Our TMN Dataset	
Total number of users	100
Total number of queries	90,851
Average number of queries per user	908.51 (±394.81)

Table 1: Query dataset statistics

2.5.3 Relative Privacy

We quantify privacy with respect to other objectives in a similar manner. One important remark here is that the above two metrics quantify what we call as "absolute" privacy obtained from the obfuscation mechanism. In fact, there is an inherent randomness in the browsing behavior of each user itself without any obfuscation. It is important to capture the additional relative value of privacy and randomness added by the obfuscation mechanism. We call this the "relative" privacy added by the obfuscation mechanism and is quantified as follows for a given metric. Consider an adversary that applies a linkage attack on S_O even though there was no obfuscation applied (i.e. $S_O = S_U$) to obtain partition $C(S_U, L, k)$. Now, comparing the privacy metrics on the output of linkage attacks $C(S_O, L, k)$ and $C(S_U, L, k)$ allows us to capture the "relative" privacy offered by the obfuscation for a given metric. More precisely, we compare S_O and S_U by computing the false negative (positive) metric as the fraction of query pairs that are (are not) in the same partition in S_U but are not (are) in the same partitions in S_O. This captures the relative error introduced to the clustering attack due to obfuscation, and it allows to compare different obfuscation mechanisms in a fair manner by removing the effect of the user's profile on her privacy.

3. USER SEARCH BEHAVIOUR

In this section, we model the web search behaviour of users in terms of the content they search for and the contextual information associated to their web searches. We first describe the web-search query dataset that we used for our experiments. Then, we explain how we model a query event based on its features.

3.1 Dataset

The most extensive, freely accessible, real world web search query dataset is the AOL dataset [27] from 2006. This dataset contains about 21 million queries from nearly 650,000 users during a three month period. However, only about 850 users issue more than 500 queries over these three months. To have a more realistic dataset, that reflects the behaviour of today's Internet users, we have chosen 100 highly active users from which we have between 400 to 1100 queries (800

[2] This metric has also been used in other domains, e.g., in location privacy [33].

Feature	Description
Behavioural features	
TimeQuery	Timestamp
DayWeek	Weekday number
TimeDay	Hour of day
NumClicks	Number of landing pages clicked
Semantic features	
TFQuery	Frequency of terms in the query
TFLandingPage	Frequency of terms in landing pages
NumQueryTerms	Number of terms in the query
NumQueryChar	Number of characters in the query
TFQueryAdult	Frequency of adult terms in the query
TFLandingPageAdult	Frequency of adult terms in the landing pages
NumSpellingErrors	Number of misspelled terms
TopicODP	Set of ODP categories of the top 8 result pages
CitiesQuery	Cities mentioned in the query
CountriesQuery	Countries mentioned in the query
TFURL	Keywords in URLs of the top 8 result pages
QueryTermPopularity	Frequency of the query terms in AOL dataset

Table 2: Features extracted from the query events

to 1000 for most of the users). These users on average issue about 8 queries per day. Table 1 presents some statistics about these datasets. Note that while our analysis relies on a much smaller set of users than the original AOL dataset, we show in the Appendix that 100 users are representative enough to estimate the overall distribution of the attack performance, and hence to quantitatively compare the effectiveness of different obfuscation mechanisms.

Every query is associated with a unique user identifier. In addition to the query terms, the dataset contains the exact time at which the user submitted the query and possibly the link of the webpage clicked by the user on the search result page, referred to as the landing page. As the landing pages might not exist anymore due to the dynamic nature of the web, we simulated the AOL queries on Google search engine and populated the dataset with up-to-date content regarding the landing pages. We note that this approach has some practical limitations. For technical reasons, our web crawler failed to retrieve some landing pages for which we could not get the page full content. In our dataset, we ignore the queries associated with such landing pages. Therefore, we only consider queries where we have the full set of features.

In order to gather realistic TMN data, we developed a Firefox plugin which takes real user queries from 100 users of the AOL dataset and issues them to the Google search engine, as a real user would do. In parallel, we have been running TMN in the browser and recorded the generated obfuscation queries. In order to simulate the three months period of the AOL dataset in a reasonable amount of time, we ran TMN during about one week and then scaled up the timestamps of the queries in order to match the three months period. The number of fake TMN queries are selected such that they are approximately equal to the number of user's real queries. Table 1 presents some statistics about the TMN dataset.

Query comparison	Description
Behavioural feature similarities	
D_Time	Difference of timestamps
S_WeekWeekend	Whether both in weekend or weekday
S_SameDaytime	Same 2 hour window of a day
D_NumberClicks	Difference of clicked landing pages
Semantic feature similarities	
S_QueryTerms	JaccardC of the query terms
S_LandPagTerms1	JaccardC of landing pages
S_LandPagTerms2	TFIDF of landing pages
S_LandPagTerms3	TFIDF of landing pages symmetric
D_QueryTermLen	Difference of query terms len
D_QueryCharacterLen	Difference of query characters len
S_AdultTerms1	JaccardC of query adult terms
S_AdultTerms2	Both queries have adult terms
S_AdultTerms3	Queries adult terms bool difference
S_LandPagAdultTerms	JaccardC of landing page adult terms
D_SpellingErrors	Difference of spelling errors
S_SpellingError1	Both queries have spelling error
S_SpellingError2	Queries spelling error bool difference
S_Level2Cat	Same ODP level 2 category
S_Level3Cat	Same ODP level 3 category
D_TreeDistance	Average ODP tree distance
S_City	Queries have same city
S_Country	Queries have same country
S_Location1	Both queries have location terms
S_Location2	Queries location info bool difference
S_UrlTerms	JaccardC of top 8 landing page URLs
D_QueryTermWeight	Difference in query term weights
D_EditDistance	Levenshtein distance of queries

Table 3: (Similarity/Difference) Relations between queries

3.2 Features

We extracted 16 features from the queries as listed in Table 2. The features can be grouped into *semantic* features and *behavioural* features [36]. For instance, features such as the time of query, day of week or time of day are features describing the behaviour of the user. Moreover, the number of clicks per query characterizes how many times usually a user clicks on a link on the result page(s).

For the *semantic* features, we extract some features that are used in Natural Language Processing (NLP). We compute the term-frequency and also the topics associated with the result pages obtained according to the Open Directory Project (ODP), an openly available hierarchical ontology [2]. When processing the terms in the query and the result pages, we stem the terms by applying the Lancaster stemmer [3].

We use ODP for categorising queries into different semantic categories/topics [10]. The ODP dataset contains about 4.2 million web-sites. The categories are organized within a tree, having the root as common top category. There are about 600,000 categories as the leaves in the ODP dataset. We categorize a query into one or multiple ODP categories, in the following way. Given a query, we retrieve the top 8 landing pages. For each landing page URL, we search the exact URL in the ODP dataset (e.g., www.domain.com/abc). If we do not find any match, we extract the domain of the URL and search for the categories associated with the domain in the ODP dataset (e.g. www.domain.com). There might be multiple categories associated with a URL. Having a total of 73,984 unique queries across 100 users, we found

Feature relation	Importance
D_Time	100
D_QueryTermWeight	24
S_UrlTerms - Jaccard	22
D_EditDistance	20
S_LandPagTerms2	17
S_LandPagTerms1	16
S_LandPagTerms3	15
D_QueryCharacterLen	13
S_QueryTerms	11
D_QueryTermLen	8
D_TreeDistance	7
D_NumberClicks	5
S_SameDaytime	4
D_SpellingErrors	4
S_LandPagAdultTerms	4
S_SpellingError1	1

Table 4: Relative importance of the features in linkage function L^{USR}, i.e. linkage function learned for attack against an obfuscation mechanism using queries from another user.

Feature relation	Importance
D_QueryTermWeight	100
D_Time	56
D_EditDistance	32
D_TreeDistance	31
S_LandPagTerms3	26
S_LandPagTerms1	20
S_LandPagTerms2	19
D_NumberClicks	18
D_QueryTermLen	14
S_UrlTerms - Jaccard	12
D_QueryCharacterLen	10
D_SpellingErrors	7
S_SpellingError1	6
S_Level2Cat	4
S_SpellingError2	4
S_LandPagAdultTerms	4

Table 5: Relative importance of the features in linkage function L^{TMN}, i.e. linkage function learned for the attack against an obfuscation using autogenerated queries (TMN).

at least one ODP category for 73,391 queries, accounting for 99.2% of the queries.

The queries may contain sensible and identifying information about the users. This includes the geographic information such as the name of a city or country, or whether the queries contain particular terms such as adult terms. We extract location information from the queries by searching the query terms in a city/country dataset [4]. If the query only contains city information, we match the city to the country as well. We also retrieve a list of adult terms and tagged each query with its respective adult terms, if existing. Moreover, since the language of most queries is English, we count the number of misspelled words per query using a dictionary.

Another piece of information that we extract from the queries is the popularity of the query terms with respect to the queries that other users search. We compute how trendy/popular each term is by counting its frequency in the whole AOL dataset. Then we compute the overall popularity of a query by summing these frequencies.

3.3 Query similarities

Features extracted about each query event represent the information about a single web-search action by the user independently from her other actions. In this section, we seek to find the relation between multiple web-searches. To this end we compare the features of pairs of query events by computing the distance or the similarity between them [36]. We compare the query features in Table 2 and we derive 27 relations between them. Again, we group these relational features into *semantic* and *behavioural*. Table 3 lists these relations and their brief descriptions.

The *behavioural* relations capture e.g., the time difference between two queries or whether they are issued on the same day of the week or hour of the day. Two queries can be associated with different intentions of the user if they are issued during a weekday or the weekend. The same applies to different hours of the day. In addition, the number of clicks on the search result URLs (landing pages) might reflect the interest of the user in the topic or her overall behaviour.

In the case of semantic features, we compute the relation between a pair of queries with respect to the query terms, and the content and topics of the landing pages. We make use of Levenshtein edit distance [25] to compute the difference between the terms in two queries.[3] This distance should usually be small between queries that are issued immediately after each other, as users repeat or modify their queries to narrow down the search results. We also use the Jaccard coefficient [5] (JaccardC) to compute the similarity between the vectors of term frequencies in a query or in a document. This approximately reflects what fraction of the terms are common between two queries or documents.

In order to compute the distance between two landing pages, we also make use of the standard information retrieval techniques to compare the relative importance of the terms in a document to another. We compute the term-frequency/inverse-document-frequency (TFIDF) metric [29]. This requires us to have a representative document frequency database. Google provides an extensive dataset of unigrams from about 8 million books [6]. By preprocessing the database (e.g. filtering non-alpha terms), we extracted 3,113,114 terms with their associated document frequency. The TFIDF metric associates more importance to the terms that are specific to the document that is analyzed, rather than the terms that appear frequently in all documents.

With these different metrics, we capture various different aspects of the relation between two queries or two landing-pages. We also compute how different two queries are in terms of the topics associated to their search results. The ODP ontology enables us to derive such measures. Every category in the ontology has a path to the root. Having calculated the ODP categories of two queries, one relation metric we consider is if the queries share a common level 2 or 3 ODP category. Having the path from the root to the leaf category, we also calculate the tree distance from the leaf category associated to one query to that of the other query. Note that a query might be associated to several ODP categories. In this case, we calculate the average distance between them.

[3] The Levenshtein edit distance is a metric which counts the number of changes that needs to be performed (insertion, subtitution, deletion of characters) between two strings until they are equal. Consequently, two equal strings have a Levenshtein distance of zero.

4. LINKAGE ANALYSIS

In this section, we discuss the methodology of the linkage attacks. We build upon the adversary's prior knowledge, namely (1) the users' search behaviour model and (2) the obfuscation mechanism. The key idea is to learn a linkage function that predicts the relation between two queries; more precisely whether they are issued by the same target user or not. We use machine learning techniques to learn this linkage function which generalizes our framework for any new obfuscation mechanisms or other datasets. We use this linkage function as the basis for the linkage attack where we run a clustering algorithm to partition the set of observed queries. We quantify the error of the linkage attack which reflects the privacy level of the attacked user.

4.1 Linkage Function

The objective of the linkage attack is to identify which observed queries belong to the target user and which of them are fake (i.e. inserted by the obfuscation mechanism). Recall that the obfuscation mechanism can introduce fake queries by auto-generating them from a bag-of-words or by sampling them from other real users. Irrespective of the type of obfuscation, the key is to have a function that predicts if two observed queries are from the same user, i.e., they are linkable. To construct such a linkage function, we build upon the users' web search behaviour and their inter-similarities and dissimilarities with respect to the fake queries. In the following, we describe our methodology for learning the linkage function.

4.1.1 Training Data and Linkage Features

We assume that the adversary has access to $\mathcal{H}_G$, historic logs of web search activities for a set of users. For the case of obfuscation using queries from another user, for any pair of users u_a, u_b present in $\mathcal{H}_G$, we can consider u_a being target user whose search activity is obfuscated by using the query events from u_b. For the case of obfuscation by auto-generated queries from bag-of-words (TMN), we assume that the adversary has access to additional sets of these auto-generated queries $\mathcal{H}_F$. In this case, for any user u_a present in $\mathcal{H}_G$, we can consider u_a as target user whose search activity is obfuscated by using the query events from $\mathcal{H}_F$. Consequently, irrespective of the type of obfuscation mechanism, we can generate a labeled pairwise list of query events e_i, e_j. Hereby, we set the linkage value to 1 if the queries are issued from the target user u_a, and 0 if one query is issued by the target user u_a while the other is added by the obfuscation mechanism. We denote this assigned linkage label by $y_{i,j}$.

Given this dataset consisting of pairwise lists of query events e_i, e_j along with the assigned label y_{ij} with value of 1 or 0, we extract the features that are listed in Table 2 for each of the query events in this data. We follow to find the relation between (the features of) any two queries and their predictive power with respect to the assigned label $y_{i,j}$. Informed by these per query features, we designed a set of pairwise features that can capture the similarity or dissimilarity between two query events. We use the list of features (functions) in Table 3 to compute the relation between each pair of features of two given queries, as discussed in Section 3.

Let $l^f_{i,j}$ be the similarity between two query events e_i and e_j with respect to feature f. We can compute this similarity for every feature f given in Table 3. For all the pairs of queries e_i, e_j in our training dataset, we compute the vector of feature similarities $link(l^f_{i,j} : \forall f)$ and label this feature vector with label $y_{i,j}$. Given this training data, we learn the linkage function, denoted by $L(e_i, e_j)$ that gives a score on whether the two queries could have been issued from the same user.

4.1.2 Learning the Linkage Function

While we could apply simple heuristics on how to best combine these different similarity scores, we decided to take advantage of machine learning based regression analysis to automatically learn the weight of different features. More precisely, our goal is to construct the linkage function that can later take as input the set of feature similarities $link(l^f_{i,j} : \forall f)$, for some pair of query events e_i and e_j and outputs the linkage score.

In general, there might not be any linear relation between the features' similarities and the linkage between two queries. In fact, depending on the (dis)similarity of two queries with respect to one feature, the importance of the other features to determine the linkage of queries differs. For example, the size of the intersection set of *terms* used in two queries is of great importance when the two queries are issued within a small time window. This is because the user is narrowing down her search using almost the same search terms. However, the similarity between the *topics* of retrieved documents is very important even when two queries are distant in time. This is because the user is interested in few topics that she searches about over time. Thus, the regression function must learn the complex relation between different features in order to predict the linkage value. To this end, we make use of the Gradient Boosted Regression Trees (GBRT) technique [19, 18]. This gradient boosting algorithm produces an ensemble of small predictors as decision trees that all together form a strong prediction model. Gradient tree boosting methods for regression and classification have various advantages including their robustness against noisy data and interpretability of the learnt model (e.g., a ranked list of feature importance) [36, 34].

The linkage function is learned as a stochastic decision tree, so it captures the importance of different feature similarities in linking queries. For our example obfuscations, we generate the appropriate training data and learn the corresponding linkage functions to be used for the attack. Let L^{USR} and L^{TMN} denote the linkage functions learned for obfuscation with queries from another user and with auto-generated queries from bag-of-words, respectively. Table 4 and Table 5 present the sorted list of top feature similarities and their normalized importance for these linkage functions.Those that appear on top are more important in the sense that knowing their values provide a stronger signal in computing the linkage value, as per the training data.

4.1.3 Aggregation over multiple Linkage Functions

The framework of using the machine learned linkage function gives us a lot of flexibility to further increase the robustness of the attack. In fact, the adversary can learn multiple of these linkage functions by varying the parameters, using different data sets or even different algorithms. This further enables the adversary to operate even when he has limited knowledge of the parameters of obfuscation mechanism.

In our setting, we consider learning a set of such linkage functions from a given dataset, by generating the training data from $\mathcal{H}_G$ for various different target users u_a. Let us consider that the adversary has learned a set of r different linkage functions denoted by $\{L_1, L_2, \ldots, L_r\}$. Given these linkage functions, we can consider different ways of aggregating their output to generate the final link score. For example, one can compute some aggregate statistics (e.g. median or mean) of the linkage score outputted by this set of functions to use as the final linkage score for a pair of queries for the clustering attack on it. On the other hand, one can compute the result of clustering for every linkage function separately, and then link together two queries if majority of the clustering solutions put these queries into the same cluster.

The specific aggregation scheme that we use in our experiments is using the median of the link scores outputted by $L_j \ \forall\ j \in [1 \ldots r]$. We use median as compared to other statistics such as mean, as it is more robust to outlier values. Further, we discuss in Section 5 how using such an aggregation based linkage function can increase the robustness of the linkage attack.

4.2 Linkage Attack

Given the linkage function $L(e_i, e_j)$ learned by the adversary, we run a linkage attack on the set $\mathcal{S}_O$. The goal is to separate the query events of the target user and those added by the obfuscation mechanism. Therefore, we compute all the features of all query events in $\mathcal{S}_O$ and for each pair of queries $e_i, e_j \in \mathcal{S}_O$, we compute the pairwise similarity $L(e_i, e_j)$ between them. Having computed L_{e_i,e_j} for all $e_i, e_j \in \mathcal{S}_O$, we have a complete weighted graph where query pairs with higher weight on their connecting edges have a higher tendency to link together. We build upon this inferred weights for the graph for the linkage attack which aims to link the queries of the user together. Moreover, we split the set of queries into multiple clusters. Our objective is to maximize the intra-cluster similarities (i.e., between queries within each cluster) and minimize the inter-cluster similarities (i.e., between queries in different clusters). We make use of the CLUTO clustering toolkit for the implementation of the attack [7, 36]. Using CLUTO, we run k-means clustering optimizing the following objective function:

$$\min \sum_{i=1}^{k} n_i^2 \frac{\sum_{q \in S_i, q' \in S_O} L(q, q')}{\sum_{q, q' \in S_i} L(q, q')} \qquad (1)$$

The output of the linkage attack is a set of clusters, given by $C(S_O, L, k) = \{S_1, S_2, \ldots, S_k\}$. This output is then used to quantify the privacy of users for various different metrics, as discussed in Section 2.

5. EVALUATION

In this section, we use our quantitative methodology to evaluate the privacy of a user with respect to two example obfuscation methods that we discussed in Section 2: (TMN) method that generates fake queries from a bag of text, and (USR) method that generates fake queries by reusing all queries of another real user.

5.1 Setup

As described in Section 3.2, we use a dataset that contains queries of 100 AOL users that we protect using both TMN and USR obfuscation mechanisms. We simulate the web-search activities of each of the target users using our Firefox plug-in, and we generate their corresponding TMN fake traces. Regarding the USR obfuscation method, we select any of the 100 user traces to be used as the fake trace. We run the experiments by adding USR fake traces 20 times, each time selecting the fake trace at random from the 100 available traces. We evaluate the user's privacy by averaging it over these 20 cases.

To construct the adversary's background knowledge, we consider that the adversary can run the obfuscation mechanism and generate fake traces for selected real traces. We assume that the attacker does not necessarily need to know the details of the obfuscation mechanism, but tries to infer it. Further, we define the query history that is available to the attacker. In our experiment, we assume that the adversary does not have access to the target user's search history, i.e., $\mathcal{H}_U = \emptyset$. However, we assume that attacker has access to some queries of the AOL users other than the target user. In our experiment, we choose the first 130 queries of the 100 users. He also knows the TMN fake traces that are generated for other users. From these datasets, we learn the linkage functions.

As discussed in Section 4, the adversary does not know a priori which set of query traces from his history set $\mathcal{H}_G$ will result in a better accuracy on a target user. In addition, learning a single linkage function from all query traces in $\mathcal{H}_G$ might not necessarily result in the best accuracy. To this end, we construct 60 linkage functions $(L_1^{TMN}, \cdots, L_{60}^{TMN})$ from different sets of TMN traces, and the same number of linkage functions from different sets of the AOL traces $(L_1^{USR}, \cdots, L_{60}^{USR})$.

For each target user and obfuscation we run the linkage attack using all the linkage functions learned from other users (i.e., those L_is that are not learned from the target user in our dataset). Then, we aggregate the linkage function as described in Section 4 by computing their median value for each pair of queries. A comparison between the clustering error of adversary, using the median aggregation, and that of each individual linkage attack in our dataset shows that in 61% of the cases the aggregate linkage function performs better than 2/3 of the individual linkage functions and results in a lower error for the attacker.

To compute the query structure privacy and semantic privacy, we run the linkage attack to split $\mathcal{S}_O$ into 2 clusters. However, for computing the query structure relative privacy, we run the attack with 10 clusters to better capture the randomness of user's web-search behavior reflected in her queries. The concentration of queries in few clusters shows the user's profile in terms of a distinct combination of query features e.g., different topics of interest.

5.2 Results

Figure 2 shows the absolute as well as the relative user privacy with respect to their query structure privacy. The x-axis shows the privacy value in terms of the normalized false positive and false negative errors. The y-axis in all the plots is the cumulative fraction of user and shows the fraction of users who all gain at least the privacy level on the x-axis.

By observing the query structure privacy (left part of figure 2), we can see that TMN offers a better privacy protection than the USR obfuscation method, and this superiority is almost at the same level across the whole set of users. As we described in section 2.5 however, we also need to quantify

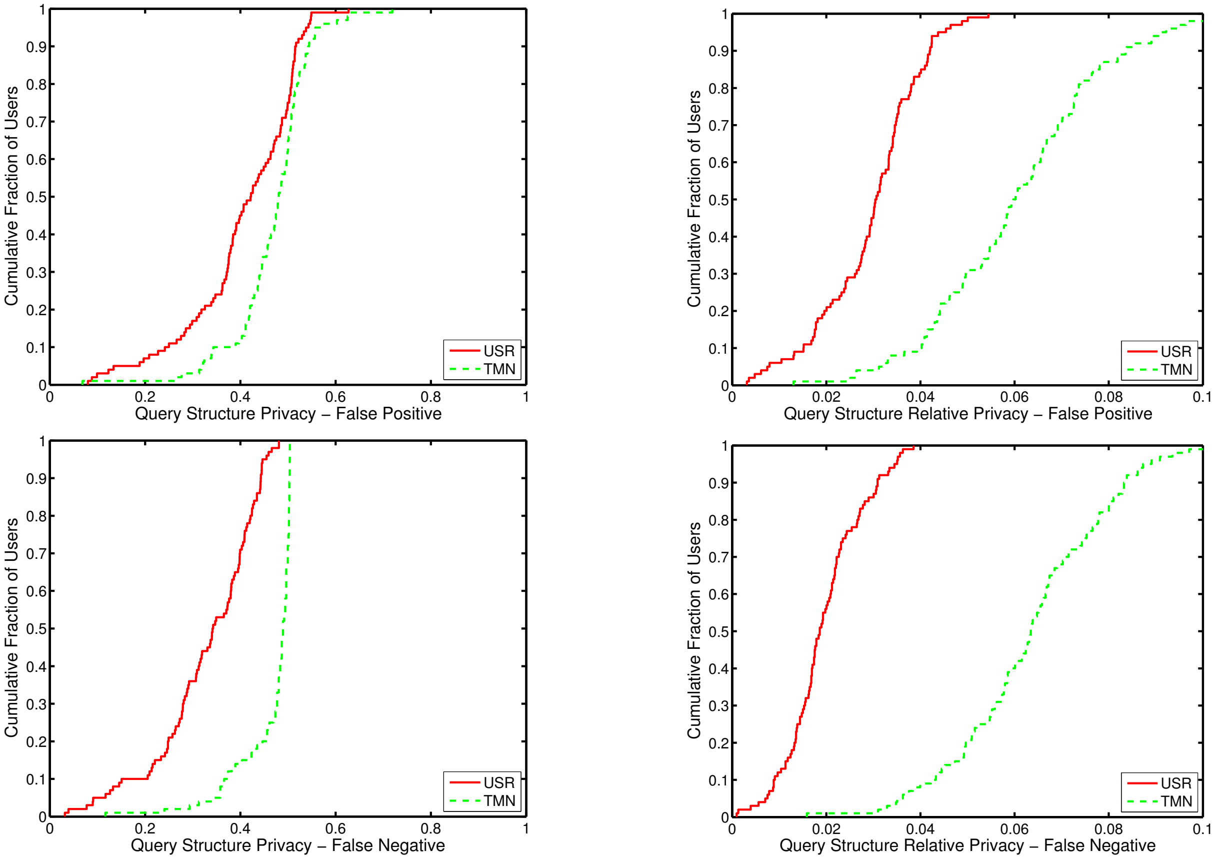

Figure 2: Empirical CDF of the target user's query structure privacy. The normalized false positive and false negative errors reflect the privacy user as to what extent the reconstructed clusters by the attacker differ from that of the user.

the privacy offered by the obfuscation mechanism after removing the inherent randomness of the user. This is done in the query structure relative privacy evaluation (right part of figure 2). Here, still TMN offers a better relative privacy than the USR obfuscation. Yet, the gap between them constantly increases as we cover a higher fraction of users, reflecting the superior effectiveness of TMN versus USR regardless of the user's behavior.

Figure 3 shows the semantic privacy of users. We compute a semantic profile as the average over the topic weights associated to the search result pages of the queries obtained from the level 2 categories of ODP. This include categories on "health, sports, news, business, shopping, recreation, science, ...". We then compute the distance between the semantic profile of $\mathcal{S}_U$ with that of $\mathcal{S}_1$ and $\mathcal{S}_2$ separately. As discussed in Section 4, $\mathcal{S}_1$ and $\mathcal{S}_2$ are retrieved from the clustering attack on $\mathcal{S}_O$. We use the cosine distance as a comparison metric between two semantic profiles. The privacy is 1 if the two profiles have no common topics, and is 0 if they have exactly the same weight on each of the topics. The attacker will later label one of the two clusters as to belong to the target user. By taking the min privacy resulted from these two clusters, we quantify the user's privacy in the worst-case scenario. In figure 3, we see that TMN offers a better overall privacy level than the USR obfuscation. On the average-case however (where we compute the mean of privacy with respect to cluster 1 and 2), the USR privacy is better for about 90% of the users. This shows the effects of user-specific background knowledge of adversary on the user's privacy. If the adversary does not have any such knowledge, any of the two clusters is not that superior to the other one as being the cluster associated with the user. However, if he has access to $\mathcal{H}_U$, he can find the cluster with higher similarity to the user's web-search log history to further break her privacy.

The USR based obfuscation uses fake queries which are sampled from real users, yet it is the TMN based obfuscation that offers better privacy across different metrics based on the results above. This is due to the (e.g., temporal and semantic) correlation between the fake queries generated using USR, across the whole query trace. In fact, one should not confuse the problem of identifying whether a set of queries are generated from a human or a machine (passing the Turing test) with the problem of linking together the queries that are generated from a human. In this paper, we are attacking the latter problem. The former problem becomes solvable only if we have specific knowledge about the behavior of the target user [8, 28], which is not our assumption here.

Given these remarks, we could design obfuscation techniques that take the best of both worlds. For example, a hybrid TMN-USR based obfuscation scheme could first

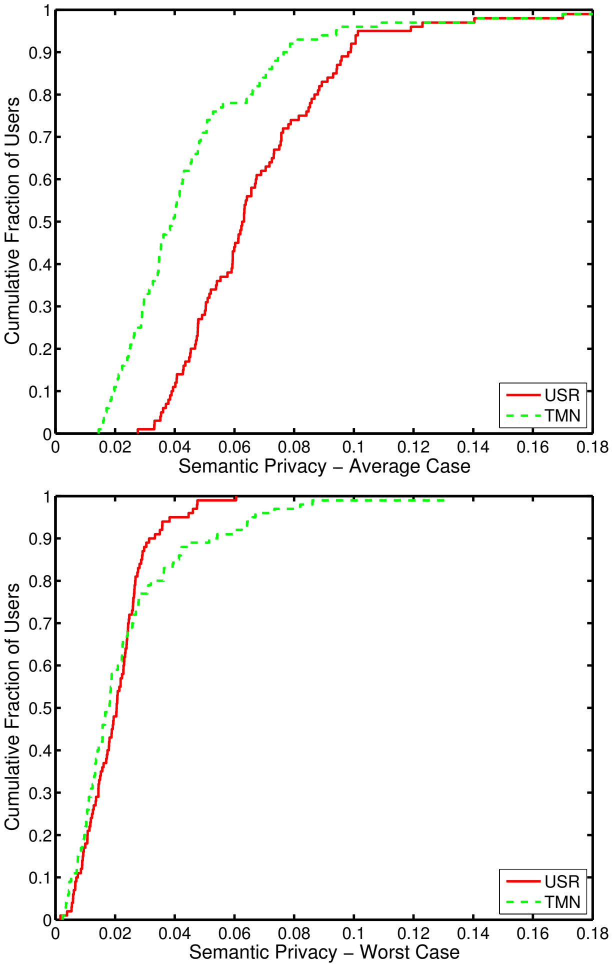

Figure 3: Empirical CDF of the target users' semantic privacy. The privacy metric is the cosine distance between a target user's true (weighted) topics of interest and the clusters constructed through the linkage attack.

take the queries from another real user, but then injects the queries and simulate the click behavior for these queries in an automated fashion, similar to what TMN does. This breaks the overall correlation between the fake queries, and make it more difficult to split them from the user's real queries.

6. RELATED WORK

A number of user-centric web search query obfuscation methods have been proposed in the literature. Theses solutions rely on the underlying idea of generating and interleaving dummy queries together with the queries of the user. The dummy queries are either sent as individual queries interleaved with the user's queries, or each user's query is modified by adding some dummy terms.

TrackMeNot (TMN) [1] consists of a Firefox plugin that issues dummy queries from predefined RSS feeds at random intervals. GooPIR [15] is a standalone application which can be used to issue queries to Google. Contrary to TMN, a real user query is extended with dummy terms and issued in a single search request. The search results are re-ranked locally based on the original search query. PRivAcy model for the Web (PRAW) [32] [17] builds an internal user profile from queries and corresponding responses. PRAW aims to issue queries which are not far from the actual user interests. Plausibly Deniable Search (PDS) [26] aims at providing k-anonymity and puts an emphasis on the fact that subsequent search queries should be related. When the user issues a query, PDS searches for a synonym query in its internal datasets and replaces the real query with a similar one. Optimized Query Forgery for Private Information Retrieval (OQF-PIR) [30] is designed to achieve perfect user profile obfuscation by making the user profile equal to the average population profile. The difference between the actual profile and the average population profile is calculated with the Kullback-Leiber divergence. Similar to OQF-PIR, Noise Injection for Search Privacy Protection (NISPP) [37] tries to find the optimal dummy query distribution among a finite number of categories. However, NISPP employs the mutual information as a metric between the observed and the real user profile. There are also some mechanisms that rely on third parties to protect user's privacy, for example, by enableing them to share queries among themselves [11].

There is no common quantitative framework and privacy metric which offer the possibility to comparing different obfuscation mechanisms. Some obfuscation solutions use information theoretic metrics to compare the observed profile and the real user profile, while other obfuscation mechanisms (such as TMN [21]) do not employ any metric. Few evaluation methods are proposed to quantify user-centric web search query obfuscation mechanisms. In [9], the authors perform a qualitative analysis by investigating the specificities of the mentioned obfuscation solutions and elaborated at least one countermeasure on how each obfuscation mechanism can be defeated. This study helps us to understand the limitations of the proposed obfuscation mechanisms, but does not allow us to quantitatively reason which one of the solutions is better than the other. In [8], the authors analysed TMN dummy queries by clustering queries and labeling them according to their similarity with the set of recently issued queries by the user. However, the clustering algorithm computes the similarity between queries irrespective to the obfuscation mechanism. So, fake queries that are similar to user queries can easily fall into the same cluster. Furthermore, [12] presents two features that can help differentiating TMN from real user queries. Another study [28] of TMN presents that simple supervised learning classifiers with few features can identify TMN queries reliably when having access to recent user search history. This work also proposes a clustering attack. However, it cannot distinguish user and TMN queries, due to focusing on the similarity between queries rather than their linkability (as we learn and use in this paper).

Our proposed linkage function does consider all differentiating features and learns their importance automatically. So, for example, the distance in time between queries influence our decision making on linking queries by looking at the value of other features rather than ignoring the queries. Our generic framework complements the existing work by proposing a systematic and quantitative approach which does not focus on any particular obfuscation mechanism, and can evaluate privacy of user even if adversary does not have any specific model on the target user.

Another related area of research which solves a similar problem is the protection of the privacy of web query logs. A survey of different obfuscation techniques for search query logs is presented [13]. In [23], the authors propose to solve the problem of releasing web query logs using differential privacy.

7. CONCLUSIONS

Having a systematic methodology to reason quantitatively about users' privacy is a necessary step towards designing effective obfuscation mechanisms. In the context of web-search privacy, notwithstanding many contributions on protecting users' web-search privacy and few specific attacks on particular obfuscation mechanisms, the lack of a generic formal framework for specifying protection mechanisms and for evaluating privacy is evident. In this paper, we have raised the questions of "what is web-search privacy? and how can it be quantified, given an adversary model and a protection mechanism?" and proposed a quantitative framework to answer these questions. In this framework, we have modeled various types of adversary's knowledge as well as the user's privacy sensitivities that leads to the definition of privacy metrics. To model the obfuscation mechanisms and adversary's knowledge about the user, we have designed a function we have called the linkage function. This is the main building block of our quantification framework and helps us to reason similar to an adversary and to distinguish pairs of queries from a user from pairs of queries that have fake information. We have constructed this function in a way that does not need, yet it can incorporate, knowledge about web-search behavior of the target user. We have used this to reason how much information (whether at the query level or semantic level) about the user is still leaked through the obfuscation process. We have applied our methodology on real datasets and compared two example obfuscation mechanisms. As the follow-up of this work, we want to design web-search obfuscation mechanisms that anticipate the possibility of linkage attacks. This strategy will lead to robust protection mechanisms.

8. REFERENCES

[1] D. C. Howe, H. Nissenbaum, and V. Toubiana, TrackMeNot - available from http://cs.nyu.edu/trackmenot.

[2] Open Directory Project (ODP) ontology, available from http://www.dmoz.org/.

[3] Lancaster Stemming Algorithm, available from http://www.comp.lancs.ac.uk/computing /research/stemming.

[4] Maxmind Free World Cities Database, Available from http://www.maxmind.com/en/worldcities.

[5] Jaccard, P. (1901) Distribution de la flore alpine dans le bassin des Dranses et dans quelques regions voisines. Bulletin de la Societe Vaudoise des Sciences Naturelles 37, 241-272.

[6] Google Books - Ngram datasets, Available from http://storage.googleapis.com/books/ngrams/books /datasetsv2.html.

[7] CLUTO - Software for Clustering High-Dimentional Datasets, available from http://glaros.dtc.umn.edu/gkhome/views/cluto.

[8] R. Al-Rfou, W. Jannen, and N. Patwardhan. Trackmenot-so-good-after-all. *arXiv preprint arXiv:1211.0320*, 2012.

[9] E. Balsa, C. Troncoso, and C. Diaz. Ob-pws: obfuscation-based private web search. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 491–505. IEEE, 2012.

[10] P. N. Bennett, K. Svore, and S. T. Dumais. Classification-enhanced ranking. In *Proc. International World Wide Web Conference (WWW)*, pages 111–120, 2010.

[11] J. Castellí-Roca, A. Viejo, and J. Herrera-Joancomartí. Preserving user's privacy in web search engines. *Comput. Commun.*, 32(13-14):1541–1551, Aug. 2009.

[12] R. Chow and P. Golle. Faking contextual data for fun, profit, and privacy. In *Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pages 105–108. ACM, 2009.

[13] A. Cooper. A survey of query log privacy-enhancing techniques from a policy perspective. *ACM Transactions on the Web (TWEB)*, 2(4):19, 2008.

[14] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[15] J. Domingo-Ferrer, A. Solanas, and J. Castellà-Roca. h(k)-private information retrieval from privacy-uncooperative queryable databases. *Online Information Review*, 33(4):720–744, 2009.

[16] P. Eckersley. How unique is your web browser? In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 1–18, Berlin, Heidelberg, 2010. Springer-Verlag.

[17] Y. Elovici, B. Shapira, and A. Maschiach. A new privacy model for hiding group interests while accessing the web. In *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, pages 63–70. ACM, 2002.

[18] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[19] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38:367–378, 1999.

[20] I. Goldberg. Improving the robustness of private information retrieval. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 131–148, May 2007.

[21] D. C. Howe and H. Nissenbaum. TrackMeNot: Resisting surveillance in web search. *Lessons from the Identity Trail: Anonymity, Privacy, and Identity in a Networked Society*, 23:417–436, 2009.

[22] 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*, CIKM '07, pages 909–914, New York, NY, USA, 2007. ACM.

[23] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *Proceedings of the 18th international*

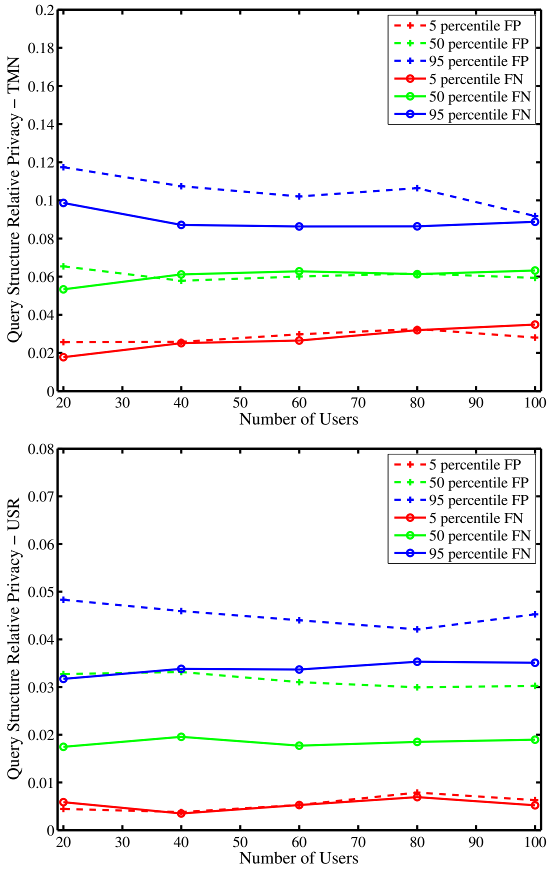

Figure 4: Effect of the size of user dataset on the quantified relative privacy.

conference on World wide web, pages 171–180. ACM, 2009.

[24] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, FOCS '97, pages 364–, Washington, DC, USA, 1997. IEEE Computer Society.

[25] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, February 1966.

[26] M. Murugesan and C. Clifton. Providing privacy through plausibly deniable search. In *SDM*, pages 768–779. SIAM, 2009.

[27] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Proceedings of the 1st International Conference on Scalable Information Systems*, InfoScale '06, New York, NY, USA, 2006. ACM.

[28] S. T. Peddinti and N. Saxena. On the privacy of web search based on query obfuscation: a case study of trackmenot. In *Privacy Enhancing Technologies*, pages 19–37. Springer, 2010.

[29] A. Rajaraman and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2012.

[30] D. Rebollo-Monedero and J. Forné. Optimized query forgery for private information retrieval. *Information Theory, IEEE Transactions on*, 56(9):4631–4642, 2010.

[31] M. K. Reiter and A. D. Rubin. Anonymous web transactions with crowds. *Commun. ACM*, 42(2):32–48, Feb. 1999.

[32] B. Shapira, Y. Elovici, A. Meshiach, and T. Kuflik. Prawa - privacy model for the web. *Journal of the American Society for Information Science and Technology*, 56(2):159–172, 2005.

[33] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux. Quantifying location privacy. In *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 2011.

[34] A. Singla, R. W. White, A. Hassan, and E. Horvitz. Enhancing personalization via search activity attribution. In *Proc. Special Interest Group On Information Retrieval (SIGIR)*, 2014.

[35] B. Tancer. *Click: What Millions of People Are Doing Online and Why it Matters*. Hyperion, 2008.

[36] R. W. White, A. Hassan, A. Singla, and E. Horvitz. From devices to people: Attribution of search activity in multi-user settings. In *Proc. International World Wide Web Conference (WWW)*, 2014.

[37] S. Ye, F. Wu, R. Pandey, and H. Chen. Noise injection for search privacy protection. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 3, pages 1–8. IEEE, 2009.

APPENDIX

The number of users that we use to learn the linkage function is a parameter in our evaluation. Here, we study how the privacy gain of using an obfuscation mechanism is affected by varying the number of users (from which we learn the linkage function). To this end, we focus on the relative privacy metric as it is the only metric that removes the effect of the target queries on the quantified privacy and only reflects the privacy gain of obfuscation. We construct the linkage function from different sets of users, of size 20 to 100, and quantify the relative privacy of TMN and USR. Each set of users includes the users in the smaller sets. For each of these cases, we can plot the empirical CDF for the users' privacy (as in Figure 2). However, to compare these plots, we adhere to some statistics (5, 50, and 95 percentiles) of these distributions. Figure 4 shows these statistics about privacy of users versus the number of users that are used for learning the linkage function. As the plot illustrates, the privacy values do not fluctuate as we change the set of users. Additionally, we observe that we do not gain in the accuracy of the privacy metric by increasing the number of users beyond 100. So, this is a reasonable size for the linkage function learning set that we use in our evaluation.