SimAttack: private web search under fire

Albin Petit, Thomas Cerqueus, Antoine Boutet, Sonia Ben Mokhtar, David Coquil, Lionel Brunie, Harald Kosch

Petit *et al. Journal of Internet Services and Applications* (2016) 7:2
DOI 10.1186/s13174-016-0044-x

Journal of Internet Services and Applications

RESEARCH Open Access

SimAttack: private web search under fire

Albin Petit[1,2*], Thomas Cerqueus[1], Antoine Boutet[1], Sonia Ben Mokhtar[1], David Coquil[2], Lionel Brunie[1] and Harald Kosch[2]

Abstract

Web Search engines have become an indispensable online service to retrieve content on the Internet. However, using search engines raises serious privacy issues as the latter gather large amounts of data about individuals through their search queries. Two main techniques have been proposed to privately query search engines. A first category of approaches, called *unlinkability*, aims at disassociating the query and the identity of its requester. A second category of approaches, called *indistinguishability*, aims at hiding user's queries or user's interests by either obfuscating user's queries, or forging new fake queries. This paper presents a study of the level of protection offered by three popular solutions: Tor-based, TrackMeNot, and GooPIR. For this purpose, we present an efficient and scalable attack – SimAttack – leveraging a similarity metric to capture the distance between preliminary information about the users (i.e., history of query) and a new query. SimAttack de-anonymizes up to 36.7 % of queries protected by an unlinkability solution (i.e., Tor-based), and identifies up to 45.3 and 51.6 % of queries protected by indistinguishability solutions (i.e., TrackMeNot and GooPIR, respectively). In addition, SimAttack de-anonymizes 6.7 % more queries than state-of-the-art attacks and dramatically improves the performance of the attack on TrackMeNot by 23.6 %, while retaining an execution time faster by two orders of magnitude.

Keywords: Privacy, Web search, Unlinkability, Indistinguishability

1 Introduction

Search engines (e.g., Google, Bing, Yahoo!) have become the preferred way for users to find content on the Internet. However, by repetitively querying for a large number of topics and websites, users disclose a large amount of personal data to these search engines. Consequently, the latter are able to create accurate knowledge on users by extracting their personal interests from their queries. Even though all user queries are not related to sensitive topics, this automated data processing about individuals raises a serious privacy issue, as users cannot control the use of their personal data and have no right to be forgotten. To deal with this issue, many solutions have been proposed to enforce private Web search. These solutions can be mainly classified into two categories. The first one, called *unlinkability*, consists in hiding the user's identity from the search engine (typically her IP address). Anonymous communication protocols (e.g., Onion Routing [1], TOR [2], Dissent [3, 4], RAC [5]) are the main solutions enforcing this property. The second type of solutions, called *indistinguishability*, aims at either altering the user's queries or hiding the user's interests. For instance, GooPIR [6] adds extra queries to the original query while TrackMeNot [7] sends periodically fake queries.

Despite these solutions improve the user privacy, a previous study [8] using a machine learning algorithm and preliminary information about the user (i.e., part of its query history) shows that an adversary is able to break both categories of solutions. However, this study was conducted using only 60 specific users (i.e., users who issued queries with a given number of keywords or queries considered as *sensitive* by the authors) and considering non-active users (called "other user" in the study). Consequently, it is not clear if an adversary is still able to break these unlinkability and indistinguishability solutions for active users. As active users can expose more information to the adversary, they represent the most difficult category of users to protect.

To better understand the limits of unlinkability and indistinguishability solutions on individual's privacy, we present in this paper a study of private Web search solutions focusing on active users. This study is conducted

*Correspondence: albin.petit@insa-lyon.fr
[1]Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621 Lyon, France
[2]Universität Passau, Innstrasse 43, 94032 Passau, Germany

with SimAttack, an efficient attack that leverages a similarity metric to capture the distance between a query and user profiles. These user profiles gather preliminary information about the users collected by the adversary. While the original version of SimAttack [9] was designed for a specific target, this paper presents a generalization of this attack for unlinkability and indistinguishability solutions.

We exhaustively evaluated our new SimAttack on three popular solutions: Tor-based, TrackMeNot, and GooPIR. Our experiments used real world Web search datasets involving up to 15,000 users. We show that SimAttack scales particularly well with respect to the number of users considered in the system. More precisely, compared to the previous machine learning attacks, SimAttack divides by 158 and 100 the execution time considering respectively 1,000 users protected by an unlinkability solution and 100 users protected by TrackMeNot. Moreover, SimAttack succeeds to de-anonymize as many users queries as the machine learning attack for unlinkability solutions, and identify up to 45.3 % of initial queries for TrackMeNot.

Finally, the generic nature of SimAttack based on a similarity distance between pre-built user profiles and a query allows an adversary to design attacks for others private Web search solutions.

For instance, we leverage SimAttack to evaluate the privacy protection offered by GooPIR. We succeed to identify at least 50.6 % of initial queries protected by this solution even if they were protected by 7 fake queries. Last but not least, as we show in our study that the previous aforementioned solutions (i.e., Tor-based, TrackMeNot, and GooPIR) do not protect properly the user privacy, we also analyze hybrid private Web search solutions: GooPIR over an unlinkability solution and TrackMeNot over an unlinkability solution.

The remaining of the paper is organized as follows. In Section 2, we present the state-of-the-art approaches. In Section 3, we describe the considered adversary model. In Section 4, we detail SimAttack and how it is able to break unlinkability solutions, indistinguishability solutions and their combinations. We then present our experimental set-up in Section 5 before evaluating the robustness of unlinkability solutions, TrackMeNot, GooPIR and hybrid solutions in Section 6, 7, 8 and 9, respectively. Finally, Section 10 concludes the paper.

2 Related work

The main solutions to privately query search engines can be classified in two categories: (*i*) systems ensuring unlinkability between requesters and their queries, and (*ii*) systems guaranteeing indistinguishability of user interests. Privacy-aware mechanism can be also directly implemented on the search engine side through Private Information Retrieval (PIR) protocols.

2.1 Unlinkability solutions

One approach to protect the user privacy from a too curious search engine is to prevent the latter from identifying the real identity of users. The identity of users is tracked through multiple techniques such as the IP address, quasi-identifiers (e.g., cookies), or fingerprints (e.g., HTTP headers, set of browser plugins [10]). While quasi-identifiers can be removed as suggested in [11], a basic solution to hide the IP address consists in leveraging a Proxy [12] or a VPN [13] server as relay. This distant server forwards user queries to the search engine on behalf of the user and returns results to the user. Unfortunately, this mechanism only shifts the privacy problem from the search engine to the relay which can collect and analyze queries from users.

Anonymous networks (e.g., Onion Routing [1], Tor [2], Dissent [3, 4], RAC [5]) represents a more complex approach to prevent a third party to map a user identity to a query. Indeed, anonymous network leverages onion routing and path forwarding to route user queries through multiple nodes before reaching the search engine. However, this approach relies on either a high number of cryptographic operations (e.g., Tor-based solutions), or all-to-all communication (e.g., RAC and Dissent) which generate a costly overhead in terms of latency and network traffic. These important overheads make anonymous networks impractical for interactive tasks such as Web search.

Other techniques try to achieve the same goal using a fully decentralized architecture. For instance, [14] and [15] proposed a protocol in which users exchange their queries in a privacy-preserving way (i.e., users do not know who issued which queries) and send them on behalf of each other. As the identity of the initial requester is unknown by the search engine and the other users, the results must be broadcasted to all the users. Therefore, these solutions generate significant overheads in terms of traffic and latency.

2.2 Indistinguishability solutions

Indistinguishability solutions consist in making the search engine only able to collect inaccurate users' queries and interests. Consequently, as the users' interests cannot be truly discovered, the privacy of users is preserved. A popular solution in this category, TrackMeNot [7], periodically sends fake queries on behalf the user. The challenge in this approach is to create fake queries that cannot be distinguished from the real ones. To do so, TrackMeNot (TMN) based the generation of fake queries on RSS feeds. However, as these RSS feeds are set up by default or manually by the user, an adversary could be able

to distinguish real queries from fake ones. For instance, in [16] authors present a simple clustering attack over small time windows that enables an adversary to retrieve fake queries. Besides, other solutions adopt a similar technique: Plausibly Deniable Search (PDS) [17] generates k plausibly deniable queries which are similar to previous user queries but on different topics. Optimized Query Forgery (OQF) [18] provides a theoretical approach to generate fake queries by measuring the Kullback-Leibler divergence between the user profile and the population distribution. Noise Injection for Search Privacy Protection (NISSP) [19] (another similar approach to OQF) gives a theoretical property that optimal fake queries should respect. However, these four solutions (TMN, PDS, OQF and NISSP) might overload the network by generating a large number of fake queries.

Another possibility to achieve indistinguishability is to modify the initial query. For instance, GooPIR [6] adds to the initial query $(k-1)$ fake queries (generated using a dictionary) where all of these queries are separated by the logical OR operation in a new obfuscated query. As GooPIR's authors consider that an adversary has no background knowledge about the user, this adversary can only guess the initial query with a probability equal to $1/k$. However, we present in Section 4 an efficient attack that is able to retrieve the initial query with a high probability considering user profiles preliminary created with their past queries. Another technique called Query Scrambler (QS) [20] protects the user by sending, instead of the initial query, a set of new queries built by generalizing the concepts of the initial query. Then, by filtering all the results, QS retrieves potential results related to the initial query. However, despite similar queries and the filtering approach proposed, the accuracy of the results remains low compared to the results obtained by issuing the original query.

A new approach using multiple HTTP cookies has been proposed to protect the user privacy while keeping the search engine able to store information about the activity of users. In this solution, the search engine splits user queries in multiple group profiles according to cookie sent with the query. Nevertheless, this approach supposes that a search engine only uses cookies to identify users and does not take into consideration other elements such as the IP address, the HTTP header, or other fingerprints. Besides, by splitting queries in multiple user profiles, this method only decreases the disclosure of information.

2.3 Private information retrieval

Search engines can implement Private Information Retrieval (PIR) protocols to offer privacy preserving query service to users. For instance, [21] presents a system in which the query is broken down in multiple buckets of words and then, the user uses homomorphic encryption to retrieve search results without revealing her initial query. However, this scheme faces many limitations to be adopted in practice (i.e., costly homomorphic encryption and it requires specific implementation at the search engine side).

3 Adversary model

Users are more and more concerned about the privacy risks of querying search engines. In this paper, we analyze the robustness of popular private Web search solutions. We considered three categories of solutions: unlinkability solutions, indistinguishability solutions, and indistinguishability solutions over unlinkability solutions. In our approach, we assumed an adversary which aims to retrieve for each protected query, both the content of the initial query and the identity of the associated user. Moreover, we assumed an adversary which was able to collect preliminary information about the interests of each user in the system. This preliminary information are stored in user profile structures. Preliminary information of users can be collected in different manners, from their social networks activity, from their posts on blogs or discussions on forums [1]. In this paper, we considered as preliminary information a part of the history of query of users.

In practice, our adversary model can be seen as a search engine receiving protected queries from users who just start to adopt a private Web search solution. In this use case, the preliminary information represent the non-protected queries sent by the users to the search engine before the exploitation of a private Web search solution. Consequently, the most active users have exposed more preliminary information to the search engine through their past querying activity.

4 SimAttack

In this section, we present SimAttack, an attack against private Web search solutions. SimAttack computes a distance between an incoming query and the preliminary information collected by the adversary (i.e., user profiles). As consequence, according to this similarity distance, the adversary is able to de-anonymize the query or differentiate the fake queries from real ones. SimAttack is a user-centric attack which tries to compromise the privacy of each user independently. In this paper, we generalized the original version of SimAttack [9] for unlinkability and indistinguishability solutions.

Compared to existing attacks, SimAttack is generic and can be adapted against all types of private Web search solutions. Indeed, by defining the user profile and the considered similarity metric, an adversary can personalize SimAttack to any type of protections.

The next sections explain how the similarity between a user profile and a query is computed, and detail how SimAttack is able to break several types of protection

mechanism based on unlinkability, indistinguishability, and indistinguishability over unlinkability.

4.1 Similarity metric between a query and a user profile

We create a similarity metric $sim(q, P_u)$ to characterize the proximity between a query q and a user profile P_u. As mentioned in [22], vector space model is widely used for text representation. Thus, we model the query q as a vector where each dimension corresponds to a separate term. For each dimension, the value of the vector is either 0 or 1 (i.e., 0 means that the keyword is not used in the query while 1 means that the keyword is used in the query). Let us define a user profile P_u as a set of queries (i.e., a set of word vectors). The similarity metric $sim(q, P_u)$ returns a value between 0 and 1 where greater values indicate that the query is close to the user's profile. It is computed as presented in Algorithm 1.

Algorithm 1: Similarity metric between a query and a user profile

input: q : a query,
P_u : profile of user u (history of query issued by u),
α : a smoothing factor.

1 **for** $q_i \in P_u$ **do**
2 $\quad coef[\,i] \leftarrow 2 \cdot |q \cap q_i| \cdot \frac{1}{|q_i|+|q|}$;
3 $coef \leftarrow sort(coef)$;
4 $sim \leftarrow coef[\,0]$;
5 **for** $i \in [\![1, |P_u|]\!]$ **do**
6 $\quad sim \leftarrow \alpha \cdot coef[\,i] + (1-\alpha) \cdot sim$
7 **return** sim ;

It first computes the value $coef[\,i]$ corresponding to the Dice's coefficient [23] between the query q and the query q_i stored in P_u, the profile of user u (line 2). As defined in Section 3, this profile contains part of the history of query already issued by the user and preliminary collected by the adversary. The coefficients $coef[\,i]$ are then ranked in ascending order (line 3). The similarity metric $sim(q, P_u)$ is finally computed as the exponential smoothing of these coefficients (lines 4 to 6). Consequently, this similarity depends on the smoothing factor α that enables to change the weight given to the coefficients. This parameter α takes its value between 0 and 1. In practice, the value of α does not strongly impact the results as shown in Section 6.1. Furthermore, we consider the Dice's coefficient which gives slightly better results than other similarity metrics (e.g., cosine similarity [22], Jaccard index [24]). As shown in our evaluations, although SimAttack is faster than concurrent approaches, the time required to perform the attack must remain as short as possible. The Dice's coefficient provides a good trade off between performance against execution time compared to edit-based and more complex token-based metrics [25].

4.2 Unlinkability attack

The de-anonymization attack consists in finding the identity of the requester of a specific query. Algorithm 2 describes this attack. For each user profile P_u previously collected by the adversary, it computes its similarity with the query q (line 3). It then returns the identity id corresponding to the profile with the highest similarity. If the highest similarity equals 0 (i.e., all similarities equal 0), the identity of the requester remains unknown and the attack is unsuccessful. Otherwise, the algorithm considers the user, id, as the issuer of the query q.

Algorithm 2: De-anonymization Solutions Attack

input: q : a query,
U : set of users.

1 $id \leftarrow u_0$;
2 **for** $u_i \in U$ **do**
3 $\quad$ **if** $sim(q, P_{u_i}) > sim(q, P_{id})$ **then** $id \leftarrow u_i$
4 **if** $sim(q, P_{id}) > 0$ **then return** id **else return** $\emptyset$

4.3 Indistinguishability attack

The attack against indistinguishability solutions aims to identify initial queries among faked or obfuscated queries received by the search engine. Contrary to the previous attack, the adversary knows the identity of the user and thus tries to pinpoint fake queries by analyzing the similarity between queries and the user profile. The attack detailed in the Algorithm 3 proceeds as follow. It first determines which obfuscation mechanism is being used. More precisely, it checks if the obfuscated query q^+ contains several fakes queries separated by the logical OR operator (line 1) (i.e., behavior of GooPIR). It might appear that the logical OR operator was introduced by the user in her query (and not by the obfuscation mechanism). Nevertheless, as the user query and all fake queries have the same number of keywords, it is easy in most of cases to detect if the logical OR was introduced by the user or the obfuscation mechanism.

Let us consider the first case in which the query q^+ is composed of $k+1$ queries (i.e., the initial query and k fake queries). The algorithm extracts each aggregated query q_i from q^+ and computes the similarity metric between these aggregated queries q_i and the user profile P_u (lines 3 and 4). Then it stores the query with the highest similarity in the variable q'. If the similarity $sim(q', P_u)$ is different

from 0, the algorithm returns q' as the initial request. Otherwise, the attack fails and the initial query is not retrieved as the $(k+1)$ queries are not similar to any user profile.

On the second case (i.e., the query does not contain the logical OR operator), it distinguishes two cases: if the adversary has a prior knowledge about RSS feeds used by the user to generate the fake queries or not. If we consider first that the adversary does not have this external knowledge, it evaluates if the similarity between the query q^+ and the user profile P_u is greater than a given threshold δ. If so, then q^+ is considered as a real query, and is therefore returned (line 8). Otherwise, the query is considered to be a fake query (line 11).

Algorithm 3: Indistinguishability Solutions Attack

```
   input: q+ : a query,
          Pu : a user profile,
          δ : a threshold,
          PFQ : a profile of fake queries.
1  if q+ = q0 OR ... OR qk then
       // q+ contains fake queries (i.e.,
          GooPIR)
2      q' ← q0 ;
3      for qi ∈ q+ do
4          if sim(qi, Pu) > sim(q', Pu) then q' ← qi
5      if sim(q', Pu) > 0 then return q'
6  else
       // q+ is either a fake query or a
          real query (i.e., TrackMeNot)
7      if PFQ = ∅ then
8          if sim(q+, Pu) > δ then return q+
9      else
10         if sim(q+, Pu) > sim(q+, PFQ) then return
           q+
11 return ∅ ;
   // q+ is a fake query
```

Conversely, if we consider the situation where the adversary knows the RSS feeds used by the user to generate the fake queries, the adversary generates fake queries using these predefined RSS feeds. These fake queries are stored in a profile P_{FQ} (same structure as a user profile P_u). Then, the adversary uses this external knowledge to distinguish fake queries (line 10). It first compares the similarity between the query q^+ and the user profile P_u (i.e., $sim(q^+, P_u)$) against the similarity between the query q^+ and the profile of fake queries P_{FQ} (i.e., $sim(q^+, P_{FQ})$). If $sim(q^+, P_u)$ is greater than $sim(q^+, P_{FQ})$, q^+ is closer to the user profile than the profile of fake queries. Consequently, q^+ is considered as a real query, and is then returned. Otherwise, the query is considered to be a fake query (line 11).

4.4 Indistinguishability over an unlinkability solution attack

The attack that breaks an indistinguishability solution over an unlinkability solution combines the two previous attacks. The attack aims at identifying both the initial requester and the initial query. To achieve that, it follows the Algorithm 4. As the attack presented in Algorithm 3, the Algorithm 4 first determines which obfuscation mechanism is being used by looking for logical OR operators (line 1). In that case, it first extracts the $(k+1)$ queries q_i from q^+ and then retrieves for each query q_i, its potential requester $id[i]$ by invoking Algorithm 2 (lines 2 to 3). Then, it removes queries which are not associated to a potential requester (lines 5 to 6), i.e. queries for which Algorithm 2 was unsuccessful. We denote the set of indexes corresponding to the remaining queries by I. Finally, if I contains one element a (i.e., only one query is associated to a potential requester), it returns the pair $(q_a, id[a])$ corresponding to the initial query q_a and to the initial requester $id[a]$ (lines 7 to 9).

However, if I contains at least two elements, it retrieves the pairs $(q_a, id[a])$ and $(q_b, id[b])$ which have the highest similarity over I and evaluates the difference between them (i.e., $sim(q_a, P_{id[a]}) - sim(q_b, P_{id[b]})$). To ensure a certain confidence in the results, if this difference is too small, the attack is thus unsuccessful, as the algorithm retrieves at least two pairs of query and requester, and it is not able to clearly identify the real one. However, if the difference is greater than a threshold (initialized at 0.01 by default), it returns the pair $(q_a, id[a])$ corresponding to the initial query q_a and to the initial requester $id[a]$ which maximizes $sim(q_a, P_{id[a]})$ over I (lines 10 to 14).

When queries do not contain OR operators, the algorithm first retrieves the potential requester *id* by calling the Algorithm 2 (line 16). If this *id* is not empty (i.e., if the attack made by the Algorithm 2 is successful), it distinguishes two cases depending if the adversary has a prior knowledge about RSS feeds used by the user. As mentioned in the previous section, if the adversary is able to generate fake queries, she creates a profile P_{FQ} (similar to user profile P_u) that contains a set of fake queries. Let us consider the first case in which the adversary does not have this knowledge (lines 18 to 19). The adversary is able to distinguish between fake queries and real ones by comparing the similarity between the query q^+ and the user profile P_{id} (i.e., $sim(q^+, P_{id})$) with the threshold δ. If $sim(q^+, P_{id})$ is greater than δ, the query is considered as a real query sent by the user *id* and thus, the pair (q^+, id) is returned.

Now, if we consider that the adversary is able to generate a set of fake queries (lines 20 to 22). The algorithm determines if the similarity distance between the query q^+ and the user profile P_{id} (i.e., $sim(q^+, P_{id})$) is greater than the similarity metric between the query q^+ and the profile of fake queries P_{FQ} (i.e., $sim(q^+, P_{FQ})$). In that case, the pair (q^+, id) is respectively considered as the initial query and the initial requester and returned by the algorithm. Otherwise, as no pair has been returned, the attack is either unsuccessful or the query is considered as a fake query (line 23).

Algorithm 4: Indistinguishability over unlinkability Attack

input: q^+ : a query,
U : set of users,
δ : a threshold
P_{FQ} : a profile of fake queries.

1 **if** $q^+ = q_0$ OR ... OR q_k **then**
  // q^+ contains fake queries
2 **for** $q_i \in q^+$ **do**
3 $id[i] \leftarrow Algorithm\_2(q_i, U)$;
4 $I \leftarrow [\![0, k]\!]$;
5 **for** $i \in [\![0, k]\!]$ **do**
6 **if** $id[i] = \emptyset$ **then** $I \leftarrow I \backslash \{i\}$
7 **if** $|I| = 1$ **then**
8 $a \in I$;
9 **return** $(q_a, id[a])$;
10 **else if** $|I| > 1$ **then**
11 $a \leftarrow$ index s.t. $sim(q_a, P_{id[a]})$ is maximal over I;
12 $b \leftarrow$ index s.t. $sim(q_b, P_{id[b]})$ is maximal over $I \backslash \{a\}$;
13 **if** $sim(q_a, P_{id[a]}) - sim(q_b, P_{id[b]}) > 0.01$ **then**
14 **return** $(q_a, id[a])$;
15 **else**
  // q^+ does not contain fake queries
16 $id \leftarrow Algorithm\_2(q^+, U)$;
17 **if** $id \neq \emptyset$ **then**
18 **if** $P_{FQ} = \emptyset$ **then**
19 **if** $sim(q^+, P_{id}) > \delta$ **then return** (q^+, id)
20 **else**
21 **if** $sim(q^+, P_{id}) > sim(q^+, P_{FQ})$ **then**
22 **return** (q^+, id)
23 **return** $\emptyset$; // the attack is unsuccessful or q^+ is a fake query

5 Experimental set-up

In this section, we provide the experimental set-up of our evaluation: the datasets, an overview of the considered indistinguishability solutions (i.e., TrackMeNot and GooPIR), and both the evaluation metrics and the concurrent approaches we use to assess the performance of SimAttack. All our experiments were conducted on a commodity desktop workstation with a 2.2 GHz quad core processor with 8 GB of memory.

5.1 Web search dataset

To evaluate the robustness of private Web search solutions, we use a real world Web search dataset from AOL Web search logs [26] published in 2006. AOL dataset contains approximately 21 million queries formulated by 650,000 users over three months (March, April and May of 2006). As this dataset contains many inactive users (i.e., users that issued too few queries), we first filtered the whole dataset to target active users. More precisely, we select users that: (*i*) sent queries on at least 45 different days (i.e., half of the dataset period), and (*ii*) issued queries on a period of at least 61 days (i.e., two-thirds of the dataset period). Finally, after this filtering phase, our dataset gathers 18,164 users who issued from 62 queries to 3,156 queries over the dataset period.

We then focus on the most active users as they are the most exposed to an adversary. We create 5 datasets containing different number of active users (from 100 to 1,000 users): `AOL100`, `AOL200`, `AOL300`, `AOL500`, `AOL1000`. To do that, we order the 18,164 users according to the number of queries they issued and then select the top 1,000 users to create the dataset `AOL1000`. We then generate the 4 other datasets as a subset of `AOL1000`. To retain similar statistical properties and ensure that users issued a significant number of queries, we generate these 4 datasets by choosing randomly (according to the distribution of the number of queries per user) the desired number of users (e.g., 100 users for `AOL100`). Figure 1 depicts the Cumulative Distribution Function (CDF) of

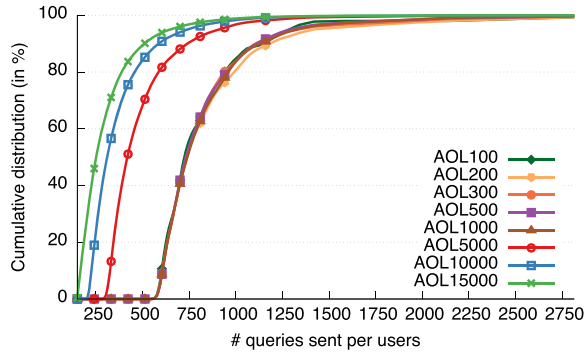

Fig. 1 Distribution of the number of queries issued per user for the different dataset

the number of queries issued by user in these 5 datasets. These CDF show that users part of these datasets issued at least 500 queries. We also note that each of these 5 datasets follows approximately the same distribution.

In addition, to assess private Web search solutions with a larger number of users, we create 3 extra datasets containing the top 5,000, 10,000 and 15,000 users: `AOL5000`, `AOL10000` and `AOL15000`. Figure 1 shows that these 3 datasets do not follow the previous distribution of queries per user due to the lack of high active users in the AOL dataset. However, results obtained with these datasets give a lower bound as having more queries in the user profile would likely increases the efficiency of the attack.

Finally, we pre-process and filter the queries of users to remove the irrelevant keywords. To achieve that, we leverage the Stanford CoreNLP library [27]. Using the tokenizer, we split queries in string vectors and then remove stop words (i.e., articles and short function words) and irrelevant keywords. Irrelevant keywords are identified with the *Named Entity Tagger* and the *Part-Of-Speech Tagger*. The former enables the recognition of names or numerical and temporal entities while the latter recognizes the function of the word. As consequence, numbers, dates or pronouns are removed. Lastly, we stem each keyword by eliminating or replacing the suffix using Porter algorithm [28].

As mentioned in Section 3, we considered that the adversary has already built a user profile for each user. Consequently, we split each dataset in two parts: a training set used to build the user profiles, and a testing set used to assess the robustness of the considered privacy-preserving mechanism. We used two third of user queries to create the training set and the remaining third of queries to create the testing set. We used two third of user queries to create the training set, and the remaining third of queries to create the testing set.

5.2 TrackMeNot

TrackMeNot, called TMN in the rest of the paper, is a Firefox plugin which periodically generates fake queries to hide user queries in a stream of related queries. After the installation of TMN, the user can define different settings to select the desired level of protection. Two main parameters impact the user protection: the RSS feed lists and the delay between two fake queries. The RSS feeds list is composed by default of four RSS feeds coming from: `cnn.com`, `nytimes.com`, `msnbc.com` and `theregister.co.uk`. The user can modify this list to remove or add extra RSS feeds. Modifying this setting is crucial, as keeping the initial list might help an adversary to distinguish between real queries and fake ones. However, it is not trivial for users to find good RSS feeds, as they should find RSS feeds that cover all their ever changing interests. Moreover, the user can customize the protection by choosing the time between two fake queries. TMN offers several possibilities: from 10 fake queries per minute to 1 fake query per hour. Consequently, the users are able to chose the quantity of noise they want to introduce in their queries. Also, the user could activate the "burst mode". In that case, when the user issues a query, TMN sends in the same time multiple fake queries to cover it.

To generate these fake queries, TMN transforms titles of articles listed in RSS feeds into queries. To do so, it randomly extracts keywords from a title and aggregate them into a fake query. The number of keywords is randomly chosen between 1 and 6. As a direct consequence, for a given title, this algorithm is able to create multiple fake queries and thus, two TMN users using the same RSS feeds do not systematically create the same fake queries.

Finally, to simulate users using TMN, we need to add fake queries to the datasets created in Section 5.1. To do that, we create our own implementation of TMN to generated the fake queries. We thus collected RSS feeds from the TMN default setting during one month and half (from August 28th, 2014 to October 9th, 2014), and we generate fake queries from the 13,878 news titles that we extracted. Additionally, we need to specify the number of fake queries that we want to generate. To do so, we consider that users used their computers 8 hours a day and have set up *60 queries per hour*. Consequently, we generate 14,880 fake queries per users (i.e., 60 queries $\times$ 8 hours $\times$ 31 days). We call `TMN100` this new dataset that contains the queries of `AOL100` plus 1,488,000 fake queries.

To ensure that the generated fake queries (built from RSS feeds captured in 2014) are using similar terms that users cared to look for in 2006, we compute the overlap between the words used in fake queries and the words used in the whole AOL dataset. We found out that 85.6 % of words used in fake queries are also contained in the AOL dataset (6,918 words out 8,082).

Furthermore, we also generate fake queries for the adversary (i.e., profile of fake queries P_{FQ} defined in Section 4.3). To do that, we generate the same number of fake queries for the adversary as for users (i.e., 14,880 fake queries).

5.3 GooPIR

GooPIR (Google Private Information Retrieval) is a Java program to query Google in a privacy-preserving way. This protection mechanism can also be used with other search engine but only Google is supported by the application. GooPIR obfuscates user queries by adding extra fake queries separated by the logical OR operation. GooPIR uses a dictionary to generate these fake queries. It can exploit any type of dictionary – in the current implementation news articles from WikiNews are used but GooPIR's authors mentioned that query logs can also be

used. By default, GooPIR creates three fake queries but users can manually set up this number.

To generate k fake queries, GooPIR selects for each keyword of the initial query, k words using the dictionary. All these k selected words have a similar usage frequency than the keyword in the initial query. Consequently, if the initial query is composed of n keywords, GooPIR selects $k \times n$ words and then creates k fake queries of n words (i.e., fake queries and user's queries have the same number of keywords).

Query answers returned by Google contain results related to the initial query but also to the fake ones. As a consequence, GooPIR implements a filtering phase that tries to remove irrelevant results introduced by fake queries. This algorithm tests for each result if its title or its description contains keywords of the initial queries. If so, the result is displayed, it is discarded otherwise.

In our experiments, to implement the behavior of GooPIR, we created the dictionary from the AOL dataset by extracting all keywords and their usage frequency from the 20 million AOL Web search queries.

5.4 Evaluation metrics

To measure the efficiency of SimAttack, we consider the precision and the recall as defined below:

$$precision = \frac{1}{|U|} \sum_{u \in U} \frac{|A_u|}{|A_u| + |\overline{A_u}|}$$

$$recall = \frac{1}{|U|} \sum_{u \in U} \frac{|A_u|}{|Q_u|}$$

where U is the set of users in the system, Q_u is the set of queries sent by user u, A_u is the set of queries issued by user u and successfully retrieved by the adversary, and $\overline{A_u}$ is specific for each solution: (i) for unlinkability solutions, $\overline{A_u}$ is the set of queries not issued by the user u but considered by the attack as sent by u ; (ii) for TMN, $\overline{A_u}$ is the set of fake queries sent by u identified by the attack as real queries ; (iii) for GooPIR, $\overline{A_u}$ is the set of queries issued by u and not retrieved by the attack ; (iv) for TMN over an unlinkability solution, $\overline{A_u}$ is either the set of fake queries issued by u identified by the attack as real queries, or the set of queries not issued by u but considered by the attack as real queries sent by u ; (v) for GooPIR over an unlinkability solution, $\overline{A_u}$ is either the set of queries issued by u and not retrieved by the attack, or the set of not queries issued by u but considered by the attack as real queries sent by u.

To evaluate the trade off between the precision and the recall, we also consider the F-Measure as the harmonic mean of precision and recall:

$$F\text{-}Measure = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

5.5 Concurrent approach

To compare the performance of SimAttack, we consider a recent attack using machine learning algorithms [8] as comparison baseline. This attack targets both unlinkability solutions and TMN, and uses Weka [29] as machine learning framework. In both cases, this attack is based on two steps: it first builds and trains a model for each user from its query history (and for TMN, it builds and trains a model from fake queries), and then it leverages these models to de-anonymize anonymous queries or to distinguish fake queries from real ones.

To de-anonymize anonymous queries, the concurrent attack uses the Support Vector Machine (SVM) classifier. This choice is motivated by a previous study [30] that shows that SVM classifier gives better results for text classification. To implement this attack, we reproduce the same condition as reported by the authors: using LibSVM (i.e., an efficient implementation of SVM), the same algorithm (i.e., C-SVC), and the same type of kernel (i.e., linear). We also let the parameter Epsilon (i.e., tolerance of termination criterion) to its default value (i.e., 0.001). However, for parameter C (i.e., cost), Weka offers a specific option (`CVParameterSelection`) to find the value that maximizes the performance of the classification. Using this option, we found out that the best value for C is 1.1.

To distinguish fake queries from real ones, the concurrent attack considers several machine learning algorithms: Logistic Regression, Alternating Decision Trees, Random Forest, Random Tree and ZeroR. For the sake of simplicity, we only use the *Logistic Regression* classifier (reported by the authors of the attack as the classifier which produces the best performance), and the SVM classifier (which was not considered in the previous study).

6 Evaluation of unlinkability solutions

In this section, we evaluate the capacity of SimAttack to compromise the anonymity of users' queries protected by an unlinkability solutions. More precisely, we assess the sensitivity of SimAttack on unlinkability solutions over various parameters. Finally, we compare the performance provided by SimAttack against the performance of the concurrent machine learning approach.

6.1 Impact of smoothing factor α

As described in Section 4.1, the smoothing parameter α of SimAttack influences the similarity distance between a query and the user profile. To measure the impact of this parameter, we report on Fig. 2 the F-Measure returned by SimAttack for varying values of α on three different datasets: `AOL100`, `AOL200`, `AOL300`. We show that α has a limited impact on the performance of SimAttack to break the anonymity of users' queries for all datasets. For instance, for `AOL100`, the F-Measure only varies from

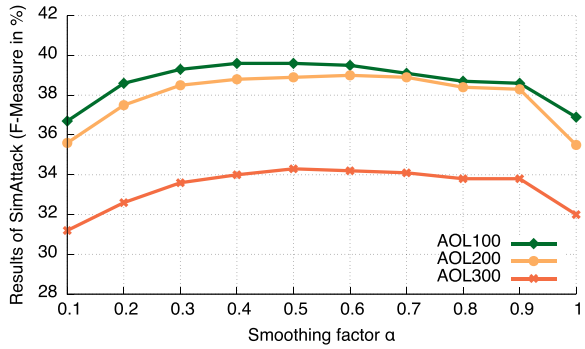

Fig. 2 The smoothing factor α has not a significant impact on the F-Measure of SimAttack

36.9 to 39.6 %. As the best F-Measure is observed for α at 0.5, we fixed the parameter α at 0.5 in the remaining evaluations.

6.2 Impact of the number of users in the system

The probability to associate a query with its correct requester is inversely proportional to the number of users in the system. Indeed, the more the users, the more the possibility to do a wrong mapping increases. To study the impact of the number of users in the system, Fig. 3 depicts both the recall and precision of SimAttack with different number of users. Results show that both recall and precision decrease when the number of users in the system increases. For instance, SimAttack de-anonymizes 36.7 % of queries with a precision of 42.9 % if we consider 100 users while it only de-anonymizes 11.9 % of queries with a precision of 17.7 % for 15,000 users. Nevertheless, this decrease is not linear and tends to stabilize as the number of users increases (around 10 % of recall for 15,000 users). As consequence, from a certain number of users, increasing this number does not offer a better protection.

6.3 Impact of targeting *p* users with the highest similarity instead of the highest one

In the previous sections, we consider that SimAttack succeeds if the attack returns the correct user. However, depending on the intentions of the adversary, this latter might consider that the attack succeeds if the initial user is among the 3, 5 or 10 most probable users. Consequently, we adapt SimAttack to link a query to the p most probable users. Results of this attack for different number of users in the system are illustrated in Fig. 4. Obviously, the number of de-anonymized queries increases according to the number of users considered by the adversary. However, this increase is rather small. For instance, the adversary de-anonymizes in average only 12.9 % more queries if the 10 most probable users are targeted compared to targeting the most probable one (i.e., $p = 10$ versus $p = 1$). In addition, if the adversary considers the 10 most probable users (i.e., $p = 10$) while the systems gather 100 users, the recall only reaches 52.8 %. This results is counter-intuitive, as the 10 most probable users represent 10 % of the dataset, this number is expected to be close to 100 % (as there is a high probability that an initial user is among the 10 most probable users). One possible explanation is that a large proportion of non-retrieved queries (74.8 % if we consider 100 users) was not retrieved because they do not contain keywords that has been already used in the previous queries of users. Therefore, these queries cannot be retrieved as their similarity with the user profile equals 0.

6.4 Impact of the number of user profiles

In the previous sections, we consider that the adversary has pre-built as many user profiles as the number of users in the system. However, in practice, the adversary might consider more user profiles than the number of users in the system. Consequently, we present on Fig. 5 the precision and the recall of SimAttack when the system gathers 100 users while the adversary considers from 100 to 1,000 user profiles. Results shows that increasing the number of

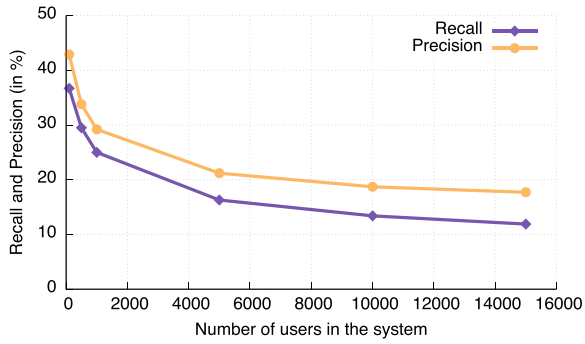

Fig. 3 Increasing the number of users in the system decreases both the recall and the precision of SimAttack

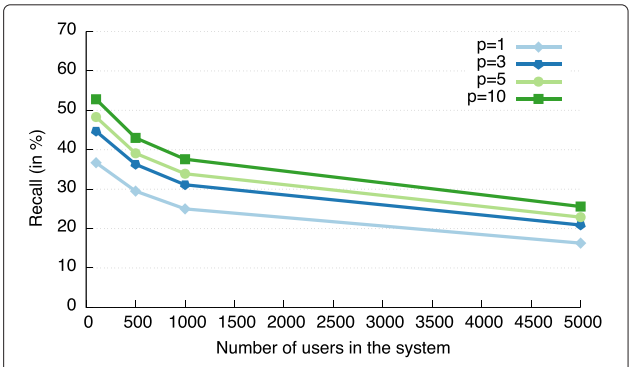

Fig. 4 The number of de-anonymized queries increases according to the number of probable users returned by SimAttack

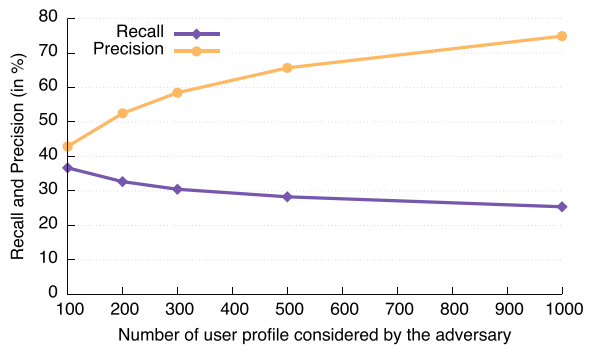

Fig. 5 Increasing the number of user profiles, with respect to the number of users in the system, decreases the precision while slightly reducing the recall

user profiles with the respect to the number of users in the system decreases the recall. For instance, adding 900 extra user profiles decreases the recall by 11.3 %. Indeed, the probability to correctly map queries from users to profile (i.e., the recall) increases as the number of possible user profiles decreases. On the contrary, the precision significantly increases according to the number of user profiles considered by the adversary. For instance, introducing 900 extra user profiles increases by 32 % the precision. Indeed, considering a larger number of user profiles reduces the number of misclassified queries between all the profiles and consequently, for a given user profile, the number of irrelevant queries (i.e., queries belonging to another user) decreases.

6.5 Impact of the size of the user profiles

The number of past queries of the users used to build the user profiles, and consequently the size of their profiles, impacts the efficiency of the unlinkability attack. To measure this impact, we depict on Fig. 6 the precision and the recall of SimAttack for `AOL100` when the profile of users pre-built by the adversary only contains a sub part of their query history (from 0 to 100 %). Results show that the efficiency of the attack decreases according to the proportion of queries considered in user profiles. Considering 100 % of past queries in user profiles provides 36.7 % of recall while considering only 5 % of queries drops this value to 16.9 %. This result illustrates that exploiting less accurate user profiles (i.e., preliminary information about the users) makes harder for the adversary to de-anonymization queries. Nevertheless, we note that the number of de-anonymized queries gently decreases (i.e., 8.7 %) when the proportion of queries considered in user profiles drops from 100 to 20 %. As a consequence, from a certain quantity of queries in a user profile, refining this profile does not significantly help the adversary to de-anonymize a higher number of queries.

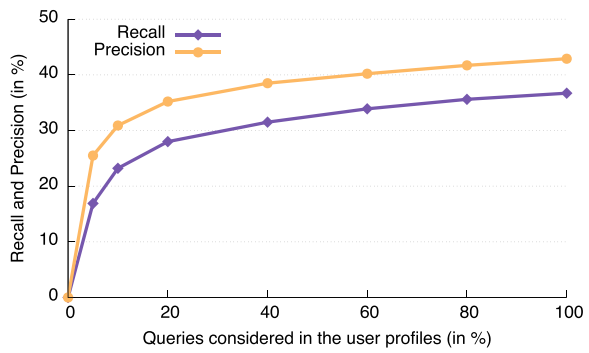

Fig. 6 The efficiency of the attack decreases according to the proportion of queries considered in user profiles

6.6 Privacy protection

We now compare the performance of both SimAttack and the concurrent machine learning attack on unlinkability solutions. Figure 7 measures for both attacks the precision and the recall for different number of users in the system. Results show that both attacks have a similar recall and precision. For instance, for AOL200, SimAttack has a recall of 36.8 % and a precision of 41.4 % against 34.4 and 39.1 %, respectively, for the machine learning attack. Nevertheless, regardless the dataset, SimAttack is slightly better than the machine learning attack. In average, the F-Measure of SimAttack is 3 % higher than the F-Measure obtained with the concurrent attack.

6.7 Scalability

We then compare the performance in term of scalability of SimAttack against the concurrent machine learning approach. More precisely, we evaluate the execution time required by these two attacks when the number of users increases. Figure 8 compares the execution time of these attacks (using a logarithmic scale). The results show that the execution time for SimAttack increases linearly with respect to the number of users while this evolution is exponential for the machine learning attack. In addition,

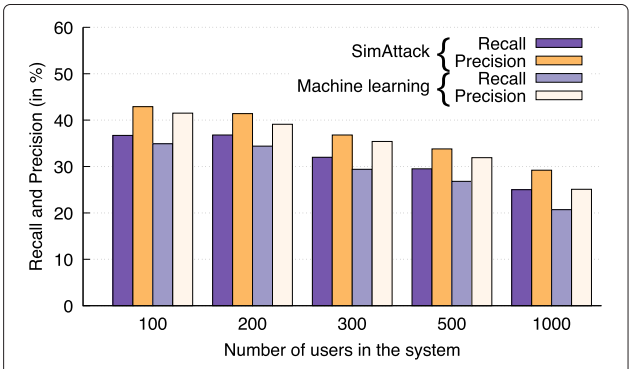

Fig. 7 SimAttack and the machine learning attack have a similar recall and precision

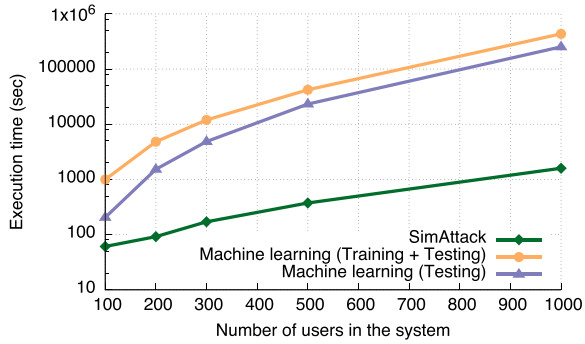

Fig. 8 SimAttack is faster than the machine learning attack especially for a high number of users

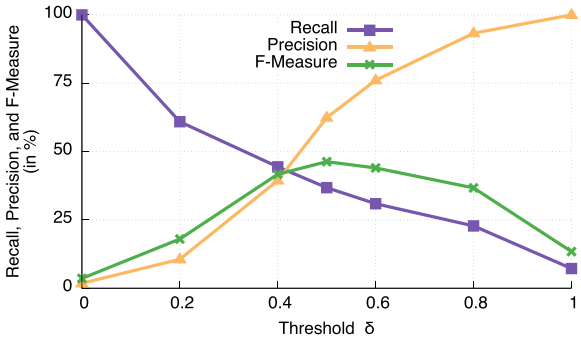

Fig. 9 The value of δ significantly impacts the performance of SimAttack

SimAttack is faster than the machine learning attack especially for a high number of users. For 1,000 users, the machine learning attack, including the time to train models, spends 434,322 *sec* while SimAttack spends 1,598 *sec*, 271 times faster. Without considering the time needed to train models (as the adversary might exploit the same models for multiple attacks), SimAttack remains faster than the machine learning (158 times faster).

6.8 Summary

SimAttack is able to de-anonymize a large number of queries protected by unlinkability solutions. Nevertheless, the SimAttack's capacity of de-anonymizing queries depends on the number of users in the system, and both the number and the quality of the preliminary user profiles collected by the adversary. While SimAttack provides similar performances than the concurrent machine learning attack, SimAttack is much more faster.

7 Evaluation of TrackMeNot

In this section, we evaluate the capacity of SimAttack to distinguish fake queries sent by TrackMeNot from the real queries sent by users. More precisely, we assess the sensitivity of SimAttack for TMN over various parameters. Finally, we compare the performance provided by SimAttack against the performance provided by the concurrent machine learning approach.

7.1 Impact of smoothing factor δ

As described in Section 4.3, the parameter δ is a threshold which controls if a query is considered as a fake query or not. Figure 9 presents the recall, the precision, and the F-Measure for several values of δ. In this attack, we consider an adversary which does not have any knowledge about the generation of fake queries (i.e., the RSS feeds used by the users). Results show that the value of δ significantly impacts the performance of SimAttack. The best results in term of F-Measure are obtained for a δ at 0.5. An adversary considering $\delta = 0.5$ is able to identify 36.8 % of initial queries with a precision of 62.4 %. Nevertheless, while the value of δ can be adjusted to achieve a higher precision, it will be however at the cost of decreasing the recall.

7.2 Impact of the external knowledge

We now analyze the robustness of TMN when SimAttack uses prior knowledge about the RSS feeds of the users to distinguish fake queries from the users' queries (as explained in Section 4.3). Table 1 lists the performances of SimAttack in term of precision, recall, and F-Measure in that case. Results show that SimAttack is able to identify 45.3 % of initial results with a precision of 87.1 %. Compared to the results obtained by SimAttack when no prior knowledge are used (i.e., in the previous section), prior knowledge increases both the recall and the precision (45.3 % versus 36.8 % for the recall, and 87.1 % versus 62.4 % for the precision). Consequently, prior knowledge and the generation of fake queries helps the adversary to break the indistinguishability offered by TMN. Keeping the default list of RSS feeds of TMN is thus not safe and can compromise the privacy of user.

7.3 Impact of the number of fake queries

As discussed in Section 5.2, users can define the number of fake queries sent by TMN. To analyze the impact of this number, we analyze the robustness of TMN against SimAttack with varying numbers of fake queries (for 1, 10 and 30 fakes queries per hour while the default value being 60). These three different settings consist in adding 7,440, 2,480, and 248 fake queries per user, respectively. Figure 10 depicts the precision and the recall provided by

Table 1 Performance of SimAttack considering an adversary with prior knowledge about RSS feeds

Recall	Precision	F-Measure
45.3 %	87.1 %	61.0 %

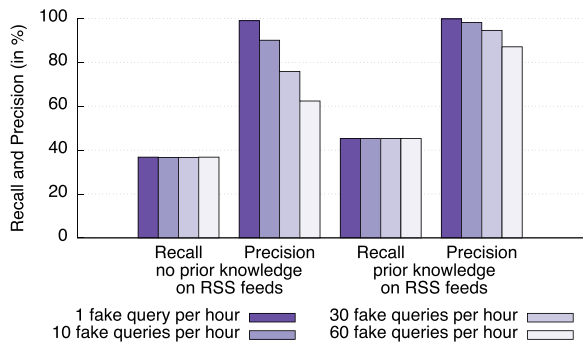

Fig. 10 The number of fake queries sent by TrackMeNot only impacts the precision of SimAttack

SimAttack for both with and without using prior knowledge. Results show that the recall remains unchanged regardless the number of fake queries (i.e., 36.8 and 45.3 % with and without using prior knowledge, respectively). Indeed, changing the number of fake queries does not affect the profile of users, and thus the same proportion of initial queries is retrieved. However, we show that the precision decreases as the number of fake queries increases. Indeed, increasing the number of fake queries increases accordingly the number of misclassified fake queries which reduces the precision. Consequently, sending too few fake queries helps the attacker to retrieve initial queries with a high precision (almost 100 % of precision for 1 fake query per hour).

7.4 Impact of the size of the user profiles

The number of queries stored in the user profiles impacts the efficiency of SimAttack. To measure this impact, we depict on Fig. 11 the precision and the recall of SimAttack for `TMN100` when the profile of users pre-built by the adversary only contains a sub part of their query history (from 0 to 100 %). Results show that exploiting smaller user profiles make harder the identification of user queries by SimAttack. For instance, if we consider

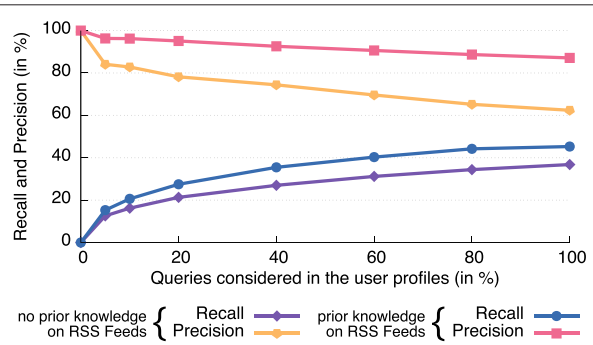

Fig. 11 Exploiting smaller user profiles make harder the identification of user queries by SimAttack

100 % of the user profiles, SimAttack identifies 36.8 % or 45.3 % of queries (depending on the exploitation or not of the prior knowledge) while this number drops to 12.6 % or 15.3 % if we consider only 5 % of the user profiles. However, the quality of the attack (i.e., the precision) increases. For instance, decreasing the number of queries considered in the user profiles from 100 to 5 % makes the precision increases from 21.7 to 92 %. Indeed, with less accurate user profiles, SimAttack does not have enough information to correctly retrieve the users. Consequently, increasing the size of user profiles increases the recall of SimAttack, but also decreases the precision as more queries get misclassified.

7.5 Privacy protection

We now compare the performance of both SimAttack and the concurrent machine learning attack on TrackMeNot using the TMN100 dataset. Firstly for the concurrent machine learning attack, Table 2 lists the precision, the recall, and the F-Measure with two different classifiers. We show that compared to the logistic regression classifier, the SVM classifier provides better performance in term of precision while achieving a slightly lower recall. Interesting enough, the SVM classifier performs better than the classifiers suggested in [8]. Secondly, as the machine learning attack requires prior knowledge on the RSS feeds, we also consider SimAttack using the same prior knowledge (results depict in Table 1). Results show that while the recall provided by SimAttack is lower than the recall provided by the machine learning attack with both classifiers, SimAttack outperforms the machine learning attacks in term of precision (87.1 % versus 29.8 % and 77.8 % for the logistic regression and the SVM, respectively). Finally, SimAttack provides better performance in term of F-Measure against the machine learning attack for both classifiers (61.0 % versus 37.4 % and 57.8 % for the logistic regression and the SVM, respectively).

7.6 Scalability

We finally compare the performance in term of scalability to conduct SimAttack on TMN against the concurrent machine learning approach. More precisely, we measure the execution time of both attacks (using prior knowledge on RSS feeds) on the TMN100 dataset. Results are reported in Fig. 12. We show that SimAttack is faster than the machine learning attack, especially for the logistic regression classifier. The training phase of the machine

Table 2 Performance of the machine learning classifiers on queries protected by TrackMeNot

Classifier	Recall	Precision	F-Measure
Logistic Regression	54.2 %	29.8 %	37.4 %
Support Vector Machine	46.0 %	77.8 %	57.8 %

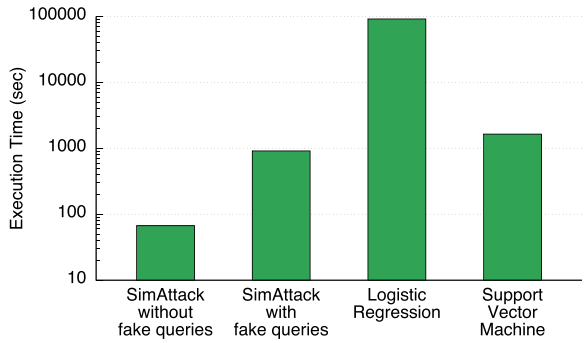

Fig. 12 SimAttack identifies user queries faster than the machine learning approach for both classifiers

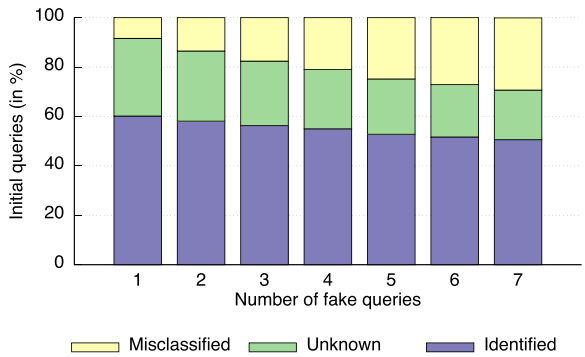

Fig. 13 The number of fake queries generated by GooPIR has a limited impact on the privacy protection of the user

learning attack is the main reason of this long execution time. Besides, using SimAttack without prior knowledge will sill reduce its execution time as SimAttack using prior knowledge spends additional time to build a profile composed of fake queries.

7.7 Summary

SimAttack succeeds to distinguish a high ratio of fake queries sent by TrackMeNot. Nevertheless, this ratio depends on the number of fake queries generated by TMN, and both the number and the quality of the preliminary user profiles collected by the adversary. Finally, SimAttack outperforms machine learning attacks and is faster.

8 Evaluation of GooPIR

In this section, we evaluate the capacity of SimAttack to distinguish fake queries generated by GooPIR. More precisely, we assess the sensitivity of SimAttack for GooPIR over different parameters.

8.1 Impact of the number of fake queries

As discussed in Section 5.3, users can define the number of fake queries generated by GooPIR. Figure 13 presents the performance of SimAttack for varying numbers of additional fake queries (from 1 to 7). Queries are classified in three categories: Identified, Misclassified and Unknown. *Identified* represents the proportion of obfuscated queries for which SimAttack retrieves the initial query (i.e., the recall), *Misclassified* represents the proportion of obfuscated queries for which SimAttack retrieves a fake query as initial query, and *Unknown* represents the proportion of obfuscated queries for which SimAttack is not able to classify any query as initial query. Results show that the number of fake queries generated by GooPIR has a limited impact on the privacy protection of the user: SimAttack retrieves 60.2 % of initial queries when only one additional fake query is generated while 50.6 % of initial queries are retrieved when 7 fake queries are generated. Besides, regardless the number of fake queries, the percentage of queries retrieved by SimAttack remains relatively high (more than half of initial queries are identified).

We then study why some queries are not identified by SimAttack (i.e., *Misclassified* and *Unknown* queries on Fig. 13). Results show that the proportion of queries in the these two categories changes according to the number of fake queries. For instance, for 1 fake query, unknown queries represent 78.9 % of non-identified queries while misclassified queries represent 21.1 %. If we consider 7 fake queries, these percentages change to 40.8 and 59.2 %, respectively.

8.2 Impact of the size of the user profiles

Similarly to the other versions of SimAttack (i.e., applied to unlinkability solutions or TrackMeNot), the size of the user profiles impacts the efficiency of SimAttack on GooPIR. To measure this impact, we assess the performance of SimAttack when the user profiles contains only a limited part of the user query history (from 0 to 100 %). Figure 14 depicts for AOL100 dataset, the percentage of

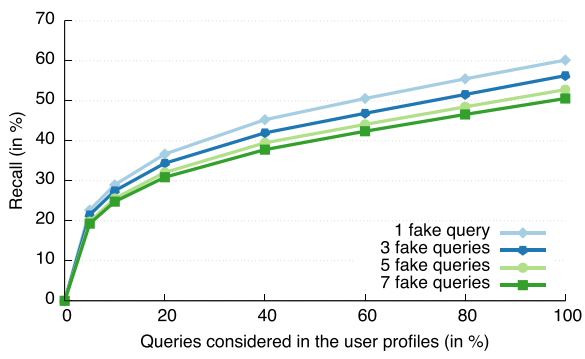

Fig. 14 GooPIR ensures a better protection if the adversary has only a smaller and inaccurate user profile

identified queries according to the ratio of the query history considered in the user profile for varying number of fake queries generated. Results show that GooPIR ensures a better protection if the adversary has only a smaller and inaccurate user profile. For instance, when 7 fake queries and only 5 % of the query history is considered in the user profiles, SimAttack retrieves 19.3 % of the initial query while this percentage increases to 50.6 % if 100 % of the query history is considered in the user profile. Indeed, SimAttack bases its identification exclusively on the user profile. Consequently, if this profile is less accurate, less user queries are identified.

Furthermore, changing the number of fake queries significantly impacts the percentage of identified queries only if the adversary considered enough queries in the user profiles. For instance, adding 6 fake queries decreases by 9.6 % the percentage of query identified when 100 % of the query history is taken into account in the user profile. This decrease drops to 3.4 % when only 10 % of the query history is considered.

8.3 Summary

SimAttack breaks GooPIR protection for more than half of the queries. In addition, the protection of the query of users is impacted by the number of fake queries: increasing the number of fake queries offers a better protection. Moreover, the size of the user profile have an impact on the performance as non-accurate user profiles make SimAttack less efficient.

9 Evaluation of indistinguishability over an unlinkability solution

As shown in the three previous sections, both unlinkability and indistinguishability approaches fail to properly protect user queries. Therefore, we carried out two further experiments which combine these two approaches (i.e., TrackMeNot and GooPIR over an unlinkability solution).

9.1 TrackMeNot over an unlinkability solution

In this section, we evaluate a solution composed of TrackMeNot over an unlinkability solution. Consequently, both queries of users and fake ones generated by TMN are sent anonymously. The remaining of this section presents a sensitivity analysis of the considered solution over various parameters.

9.1.1 Without prior knowledge on RSS feeds

We now analyze the robustness of TMN over an unlinkability solution if the adversary does not consider prior knowledge on RSS feeds of users. Figure 15 presents the recall, the precision, and the F-Measure for several values of δ on the `TMN100` dataset. Results show that δ significantly impacts the performance of SimAttack: the

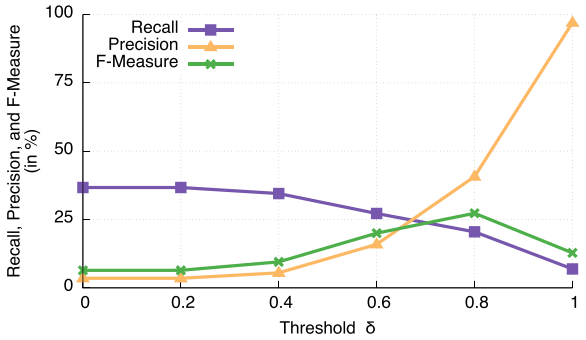

Fig. 15 The value of δ significantly impacts the performance of SimAttack

F-Measure varies from 6.4 to 27.3 % and gives it best value for δ equals 0.8. For this value, an adversary is able to identify 20.5 % of real queries with a precision of 40.7 %. Compared to the results obtained with TrackMeNot alone, adding the unlinkability solution decreases the recall and the precision (20.5 % versus 36.8 % for the recall, and 40.7 % versus 62.4 % for the precision). Indeed, as the adversary does not know the identity of the user, a lot of queries are now misclassified. In addition, compared to an unlinkability solutions alone (Section 6.2), combining TMN with an unlinkability solution decreases the percentage of queries de-anonymized by SimAttack from 36.7 to 20.5 %. Consequently, using TMN with an unlinkability solution protects 16.2 % more queries.

9.1.2 With prior knowledge on RSS feeds

We then analyze the performance of SimAttack when the adversary leverages prior knowledge on the RSS feeds of the users in order to distinguish fake queries to real queries. Table 3 lists the recall, the precision, and F-Measure obtained by the attack. Results show that SimAttack succeeds to identify 35.4 % of the queries of users with a precision of 14.7 %. Compared to the results of TrackMeNot alone, adding the unlinkability solution increases the protection of user's query, as 9.9 % more queries are not identified by SimAttack. However, the precision has significantly dropped: from 87.1 to 14.7 % without and with the unlinkability solution, respectively. Compared now to the precision obtained by SimAttack on an unlinkability solution alone (see Section 6.2), introducing the fake queries of TMN decreases the precision from 42.9 to 14.7 %. The decrease of precision is a direct result of the high ratio of fake queries misclassified.

Table 3 Performance of SimAttack considering an adversary with prior knowledge on RSS feeds

Recall	Precision	F-Measure
35.4 %	14.7 %	20.7 %

Furthermore, compared to the results obtained by SimAttack when no prior knowledge is considered (i.e., Section 9.1.1), SimAttack with prior knowledge increases by 14.9 % the recall but decreases by 16 % the precision. Overall, the F-Measure without prior knowledge is higher than the one with prior knowledge (27.3 versus 20.7 %). Interesting enough, SimAttack on TMN alone with prior knowledge provides higher performance than without prior knowledge (their F-Measures are 61.0 and 46.3 %, respectively).

9.1.3 Impact of the number of fake queries

We now evaluate the protection offered by TMN over an unlinkability solution according to the number of fake queries periodically sent by TMN. Figure 16 presents the recall and the precision for SimAttack with and without prior knowledge on RSS feeds. Results show that the number of fake queries does not change the recall (i.e., 20.5 and 35.4 % without and with using prior knowledge on RSS feeds, respectively). However, the precision strongly depends on the quantity of fake queries. Indeed, decreasing the number of fake queries from 60 to 1 fake query per hour, increases the precision by 31.4 and 32.5 % (for SimAttack without and with prior knowledge, respectively). Consequently, if the user wants a proper protection, TMN has to send a high number of fake queries.

9.1.4 Impact of the size of the user profiles

We finally evaluate how the size of the profile impacts the performance of SimAttack when TrackMeNot is combined with an unlinkability solution. Figure 17 depicts the recall and the precision of SimAttack with and without prior knowledge for a size of user profile changing from 0 to 100 %. Similarly to the other versions of SimAttack, smaller user profiles decreases the performance. For instance, decreasing the size of the user profiles from 100 to 5 % makes SimAttack able to identify 13.8 and 21.7 % less queries without and with prior knowledge on RSS feeds, respectively. In addition, results show that the precision decreases according to the size of the user profile. Nevertheless, considering from 100 to 5 % of the user profiles, the precision looses 51.2 and 15.0 %, respectively, for SimAttack with and without prior knowledge on RSS feeds of users. Furthermore, compared to the results obtained with TrackMeNot alone (Section 7.4), these results are only slightly lower, excepted for the precision of SimAttack with both prior knowledge and 5 % of the user profile, which drops from 96.3 to 14.7 %.

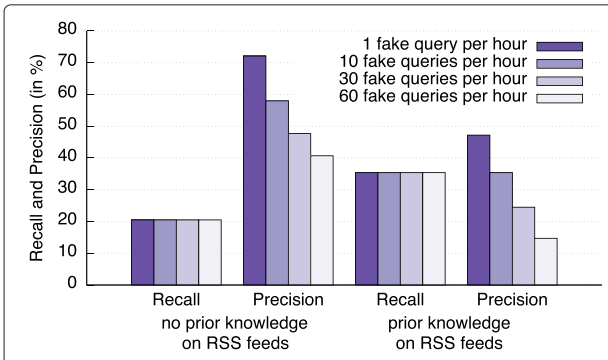

Fig. 16 The number of fake queries sent by TrackMeNot over an unlinkability solution does not change the recall but strongly impact the precision of SimAttack

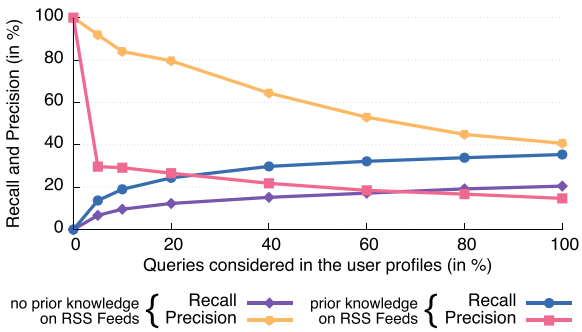

Fig. 17 Exploiting smaller user profiles make harder the identification of the user queries

9.2 GooPIR over an unlinkability solution

In this section, we assess a solution combining the obfuscation of GooPIR and an unlinkability solution. The remaining of this section presents a sensitivity analysis of the considered solution over various parameters.

9.2.1 Impact of the number of fake queries

Firstly, we evaluate the impact of the number of fake queries generated by GooPIR. As the considered solution combines GooPIR and an unlinkability solution, SimAttack needs to retrieve the user query (among the $(k + 1)$ queries) but also the identity of the user who issued the protected query (among all users in the system). As explained in Section 4.4, SimAttack returns the most probable pair $(query, user)$. Figure 18 shows the percentage of queries retrieved by SimAttack and initially protected with a varying number of fake queries (from 1 to 7). Results shows that the number of fake queries has a limited impact on the protection of queries: changing from 1 to 7 fake queries protects only 9.2 % more queries. Besides, the percentage of queries retrieved by the SimAttack remains relatively high (i.e., a recall at 27.6 % for 7 fake queries) considering that queries are protected by two independent private Web search solutions. Furthermore, compared to the results of GooPIR alone (Section 8.1), the current percentage of initial queries identified is 23 % lower (for 7 fake queries) showing that adding the unlinkability solution has a huge impact on the user's protection.

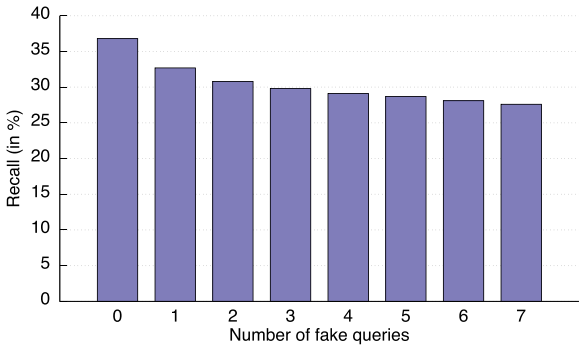

Fig. 18 The number of fake queries has a limited impact on the protection of queries

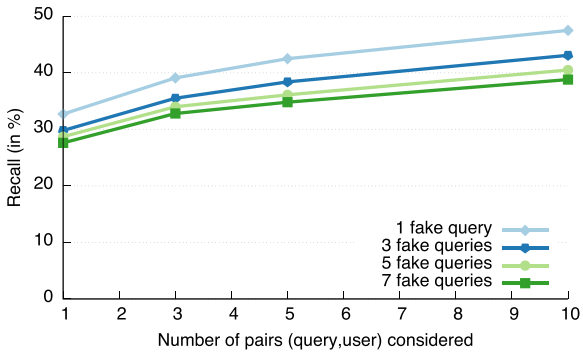

Fig. 20 The recall slightly increases according to the number of pairs (*query*, *user*) returned by SimAttack

9.2.2 *Impact of the size of the user profiles*

We then evaluate the impact of the size of the user profiles on the performance of SimAttack when GooPIR is combined with an unlinkability solution. Figure 19 presents the recall of SimAttack for different proportion of queries in the user profile (from 0 to 100 %). Results show that GooPIR over an unlinkability solution protects more strongly the queries of users if the adversary has smaller user profiles: for 1 fake query adding by GooPIR, SimAttack identifies 16.1 % of queries when 5 % of the user profile is considered, while 32.7 % of the queries are identified when 100 % of the profile is considered. Moreover, the linearity of the curve (considering the points between 5 to 100 %) means that the more information is own by the adversary, the more queries she is able to retrieve.

9.2.3 *Impact of considering more than one pair (query,user)*

In the previous sections, we consider that SimAttack succeeds if the attack returns the correct pair (*query*, *user*). However, an adversary can be interested to consider more probable pairs. We report on Fig. 20 the recall of SimAttack considering different number of pairs: from 1 to 10 pairs. Results show that the recall increases according to the number of pairs considered by the adversary. For instance, considering 10 pairs instead on 1 makes the recall increases in average by 13.4 %. Nevertheless, this recall improvement remains relatively low.

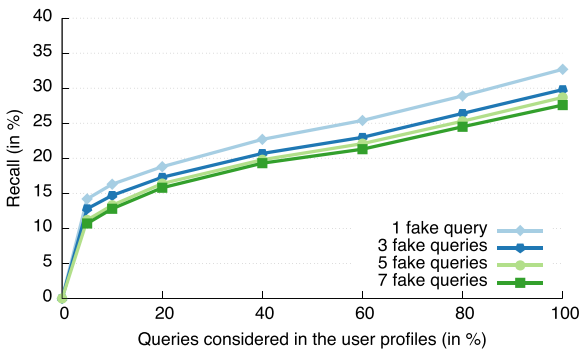

Fig. 19 Exploiting smaller user profiles make harder the identification of the user queries by SimAttack

9.3 Summary

Combining an indistinguishability technique (i.e., TrackMeNot or GooPIR) over an unlinkability solution gives a better protection to the queries of user, especially if the adversary is not able to collect a large quantity of information about the user or if the user configures its indistinguishability solution to sent a high number of fake queries. Nevertheless, in most of cases, the adversary is still able to retrieve a non-negligible proportion of user queries.

10 Conclusion

This paper presents SimAttack, a generic attack that targets popular private Web search solutions. SimAttack leverages a similarity metric to capture the distance between a query and pre-built user profiles gathering preliminary information about the user interests. We exhaustively evaluate SimAttack using a real world Web search dataset. We show that SimAttack succeeds to deanonymize, or retrieve among fake queries a high ratio of initial queries from user.

Our analysis shows that neither unlinkability solutions, nor TrackMeNot and GooPIR protects properly the users. Besides, we study the combination of TrackMeNot and GooPIR over an unlinkability solution. The first combination (i.e., TrackMeNot over an unlinkability solution) gives a satisfactory protection when enough fake queries are periodically sent. However, this solution generates an important overhead in term of message on the network. The second combination (i.e., GooPIR over an unlinkability solution) still suffers from a high ratio of initial queries identified by SimAttack.

Dynamically evaluating protected queries in order to measure their level of protection over time represents an interesting research agenda for future works. For instance,

thanks to this dynamic assessment, it will be possible to adapt the queries protection before sending them, and to reinforce the user awareness.

Endnote

[1]How the adversary collects preliminary information remains outside the scope of this paper.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
AP designed and implemented SimAttack, carried out the evaluation and wrote the manuscript. TC and SBM participated in the design of SimAttack, contributed to the definition of the experiments and revised the manuscript. AB wrote and revised the manuscript. DC, LB and HK supervised and coordinated the work. All authors read and approved the final manuscript.

Acknowledgements
The presented work was supported by the EEXCESS project funded by the EU Seventh Framework Programme FP7/2007-2013 under grant agreement number 600601. Research reported in this publication has been carried out as part of the International Research and Innovation Centre in Intelligent Digital Systems (IRIXYS).

Received: 23 July 2015 Accepted: 23 March 2016
Published online: 18 April 2016

References

1. Goldschlag D, Reed M, Syverson P. Onion routing. Commun ACM. 1999;42(2):39–41.
2. Dingledine R, Mathewson N, Syverson P. Tor: The second-generation onion router. In: Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13. San Diego, CA: USENIX Association; 2004. p. 21–1.
3. Corrigan-Gibbs H, Ford B. Dissent: accountable anonymous group messaging. In: Proceedings of the 17th ACM Conference on Computer and Communications Security. Chicago, Illinois, USA: ACM; 2010. p. 340–50.
4. Wolinsky DI, Corrigan-Gibbs H, Ford B. Dissent in numbers: Making strong anonymity scale. In: Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. Hollywood, CA, USA: USENIX Association; 2012. p. 179–92.
5. Ben Mokhtar S, Berthou G, Diarra A, Quéma V, Shoker A. Rac: A freerider-resilient, scalable, anonymous communication protocol. In: Proceedings of the 33rd IEEE International Conference on Distributed Computing Systems. Philadelphia, PA, USA: IEEE Computer Society; 2013. p. 520–9.
6. Domingo-Ferrer J, Solanas A, Castellà-Roca J. h(k)-private information retrieval from privacy-uncooperative queryable databases. Online Inform Rev. 2009;33(4):720–44.
7. Toubiana V, Subramanian L, Nissenbaum H. Trackmenot: Enhancing the privacy of web search. CoRR. 2011. http://arxiv.org/abs/1109.4677.
8. Peddinti ST, Saxena N. Web search query privacy: Evaluating query obfuscation and anonymizing networks. J Comput Secur. 2014;22(1): 155–99.
9. Petit A, Cerqueus T, Ben Mokhtar S, Brunie L, Kosch H. Peas: Private, efficient and accurate web search. In: Proceedings of the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. Helsinki, Finland: IEEE Computer Society; 2015. p. 571–80.
10. Eckersley P. How unique is your web browser? In: Proceedings of the 10th International Conference on Privacy Enhancing Technologies. Berlin, Germany: Springer-Verlag; 2010. p. 1–18.
11. Saint-Jean F, Johnson A, Boneh D, Feigenbaum J. Private web search. In: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society. Alexandria, VA, USA: ACM; 2007. p. 84–90.
12. Shapiro M. Structure and Encapsulation in Distributed Systems: the Proxy Principle. In: Proceedings of the IEEE 6th International Conference on Distributed Computing Systems. Cambridge, MA, USA: IEEE Computer Society; 1986. p. 198–204.
13. Seid HA, Lespagnol AL. Virtual private network. Google Patents. US Patent 5,768,271. 1998. https://www.google.com/patents/US5768271.
14. Castellà-Roca J, Viejo A, Herrera-Joancomartí J. Preserving user's privacy in web search engines. Comput Commun. 2009;32(13–14):1541–51.
15. Lindell Y, Waisbard E. Private web search with malicious adversaries. In: Proceedings of the 10th International Conference on Privacy Enhancing Technologies. Berlin, Germany: Springer-Verlag; 2010. p. 220–35.
16. Al-Rfou R, Jannen W, Patwardhan N. Trackmenot-so-good-after-all. arXiv preprint arXiv:1211.0320. 2012.
17. Murugesan M, Clifton C. Providing Privacy through Plausibly Deniable Search. Sparks, NV, USA: Society for Industrial and Applied Mathematics; 2009, pp. 768–79. Chap. 65.
18. Rebollo-Monedero D, Forné J. Optimized query forgery for private information retrieval. IEEE Trans Inf Theory. 2010;56(9):4631–42.
19. Ye S, Wu F, Pandey R, Chen H. Noise injection for search privacy protection. In: Proceedings of the IEEE 12th International Conference on Computational Science and Engineering. Vancouver, Canada: IEEE Computer Society; 2009. p. 1–8.
20. Arampatzis A, Efraimidis P, Drosatos G. A query scrambler for search privacy on the internet. Inform Retriev. 2013;16(6):657–79.
21. Pang H, Ding X, Xiao X. Embellishing text search queries to protect user privacy. VLDB'10. 2010;3(1–2):598–607.
22. Singhal A. Modern information retrieval: A brief overview. IEEE Data Eng Bull. 2001;24(4):35–43.
23. Dice LR. Measures of the amount of ecologic association between species. Ecology. 1945;26(3):297–302.
24. Jaccard P. The distribution of the flora in the alpine zone. New Phytologist. 1912;11(2):37–50.
25. Cohen W, Ravikumar P, Fienberg S. A comparison of string metrics for matching names and records. In: Proceedings of the KDD-03 Workshop on Data Cleaning and Object Consolidation. Washington, DC, USA: ACM; 2003. p. 73–8.
26. Pass G, Chowdhury A, Torgeson C. A picture of search. In: Proceedings of the 1st International Conference on Scalable Information Systems. Hong Kong: ACM; 2006. p. 1.
27. Manning CD, Surdeanu M, Bauer J, Finkel J, Bethard SJ, McClosky D. The Stanford CoreNLP natural language processing toolkit. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. Baltimore, MD, USA: Association for Computational Linguistics; 2014. p. 55–60.
28. Porter MF. An algorithm for suffix stripping. Program. 1980;14(3):130–7.
29. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The weka data mining software: an update. ACM SIGKDD Explor Newsletter. 2009;11(1):10–18.
30. Hearst MA, Dumais ST, Osman E, Platt J, Scholkopf B. Support vector machines. IEEE Intell Syst. 1998;13(4):18–28.