k-subscription: Privacy-Preserving Microblogging Browsing Through Obfuscation

Panagiotis Papadopoulos
FORTH-ICS, Greece
panpap@ics.forth.gr

Antonis Papadogiannakis
FORTH-ICS, Greece
papadog@ics.forth.gr

Michalis Polychronakis
Columbia University, USA
mikepo@cs.columbia.edu

Apostolis Zarras
Ruhr-University Bochum
apostolis.zarras@rub.de

Thorsten Holz
Ruhr-University Bochum
thorsten.holz@rub.de

Evangelos P. Markatos
FORTH-ICS, Greece
markatos@ics.forth.gr

ABSTRACT

Over the past few years, microblogging social networking services have become a popular means for information sharing and communication. Besides sharing information among friends, such services are currently being used by artists, politicians, news channels, and information providers to easily communicate with their constituency. Even though following specific channels on a microblogging service enables users to receive interesting information in a timely manner, it may raise significant privacy concerns as well. For example, the microblogging service is able to observe all the channels that a particular user follows. This way, it can infer all the subjects a user might be interested in and generate a detailed profile of this user. This knowledge can be used for a variety of purposes that are usually beyond the control of the users.

To address these privacy concerns, we propose *k-subscription*: an obfuscation-based approach that enables users to follow privacy-sensitive channels, while, at the same time, making it difficult for the microblogging service to find out their actual interests. Our method relies on obfuscation: in addition to each privacy-sensitive channel, users are encouraged to randomly follow $k-1$ other channels they are not interested in. In this way (i) their actual interests are hidden in random selections, and (ii) each user contributes in hiding the real interests of other users. Our analysis indicates that *k-subscription* makes it difficult for attackers to pinpoint a user's interests with significant confidence. We show that this confidence can be made predictably small by slightly adjusting k while adding a reasonably low overhead on the user's system.

Categories and Subject Descriptors

K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*; C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

Keywords

Obfuscation; Microblogging Services; k-anonymity; Anonymous subscription; Privacy-Preserving Browsing;

ACSAC '13 Dec. 9-13, 2013, New Orleans, Louisiana USA

1. INTRODUCTION

Microblogging social networking services, such as Twitter, Tumblr, Identi.ca and Fanfou, enable users to have timely access to all their information and entertainment needs. Through a publish-subscribe model, a user subscribes ("follows" in the language of microblogging) to a number of other users, or information providers in general ("channels"). These channels may correspond (i) to the user's friends, (ii) to artists the user is interested in, (iii) to politicians the user supports, (iv) to news channels, (v) to religious channels, (vi) to hospitals or doctors, and so on. When a new message is published in one of the channels a user follows, it appears on user's screen. This personalized information delivery, although useful, raises some privacy concerns. For example, if a user follows a particular politician, the service may be able to infer the user's political beliefs. If the user follows a religious channel, the service may be able to infer the user's religious preferences. If the user follows a channel about a particular health problem, the service may be able to infer that the user is interested in this health problem.

As microblogging services become increasingly popular with hundreds of million of users, such privacy concerns will become even more crucial in the future. Indeed, by providing a handy user interface and by co-locating all of a user's interests into one convenient screen, microblogging services contain a huge amount of information about users' interests and needs. In this setting, there may be users that although they really want to have access to timely information, they may not be willing to disclose their personal preferences and interests to the microblogging service, and potentially to the corporations that may collaborate with it.

To avoid being identified, users may log into the microblogging service using a pseudonym or a fake account. However, information from the user's IP address, third-party tracking cookies [15], or browser fingerprinting [8] may enable the microblogging service to pinpoint the identity of the user and trace the fake account to a real person. Although a fake account might seem to be giving a sense of pseudonymity, it will not take long for a service to gather enough information so as to correlate a fake account with a real identity. A momentary lapse of vigilance is usually all it takes from the user to provide ample identifying information to the microblogging services. Another way to protect a user from being identified by a microblogging service is to use an anonymization service to access the Internet. For example, widely-used anonymous communication networks, such as Tor [6], are effective at hiding a user's real IP address, but are of little help when the user logs into a service such as a microblogging service. Additionally, a potential blocking of Tor nodes will make it difficult, if not impossible, to use it.

One might envision to use both a fake account and Tor at the same time in order to have stronger anonymity. Unfortunately, this approach would still be subject to contamination from information gathered during different browsing sessions, such as browser fingerprints or cookies, which would alternate between anonymous and eponymous web browsing. To remedy this problem one might use (i) a fake account, (ii) an anonymization network, such as Tor, and (iii) a virtual machine per browsing session or destination web site, so as to limit cross-contamination. Although this triplet seems to provide a comfortable level of anonymity, its applicability and ease of use is questionable: it is not clear whether ordinary users will find it easy to install all the necessary software. Also, modern devices, such as tablets and smartphones, which have rapidly penetrated the market, may not be able to install virtual machines.

Numerous approaches have been proposed to conceal users' activities when browsing the web [4,5,9,16]. However, none of them can be adopted for microblogging services as the underlying threat model or technical implementation does not fit to the use case we address in this paper. Most of the previously described approaches rely on the fact that in order to conceal one's interests we need to hide one's real identity. This paper addresses a world where this is not easy, and may not even be possible. Indeed, a wide variety of web tracking mechanisms aided by increasing legal pressure on anonymity systems may lead to a world where anonymous web surfing would practically belong to the past. At this point we see two choices: (i) to believe that we will always be able to anonymously browse the web, and thus the massive losses of privacy we see in real life will never percolate to cyberspace, or (ii) to proactively develop privacy-preserving approaches for a world where it will be difficult, if not impossible, to hide one's real identity.

In this study, we explore the second choice and develop new obfuscation-based approaches to preserve privacy and conceal users' real interests. Whenever a user is interested in following an actual channel C_1, she is encouraged to follow $k-1$ noise channels as well: C_1, C_2, C_3, ..., C_k. If the user follows all these channels, the microblogging service will not be sure which channel she is actually interested in. Moreover, if there is a plethora of users following the channel C_1 while they are not actually interested in it, the microblogging service will not be able to identify the users that are interested in C_1. By fine tuning this number of channels, users are able to achieve the level of privacy they are comfortable with.

Obfuscation itself is an age-old idea. It has been used in war to confuse radars detecting incoming bomber planes [23], in consumer organizations through super market card swapping to confuse marketers wanting to build customer profiles [10], and in web searches [4,11] to hide the user's real interests from search engines. To the best of our knowledge, this is the first time that obfuscation is applied to provide privacy in the area of microblogging services. In cases where it is not possible to hide an event, such as an approaching war plane, a query to a search engine, and a channel followed in a microblogging service, obfuscation provides a reasonable mechanism to confuse the adversary to the point of not being able to distinguish real information from the added noise.

In this paper, we present the design and analysis of *k-subscription*: our approach to obfuscate users' real interests in microblogging services. We study the anonymity provided by two different obfuscation strategies in an analytical way, and we evaluate the anonymity offered by *k-subscription* in a realistic scenario using simulations. To assess the practical feasibility and effectiveness of *k-subscription*, we have implemented an extension for the Chrome browser that enables privacy-preserving subscription to Twitter channels through obfuscation. Our experimental evaluation shows that the overhead introduced by *k-subscription* is reasonable in practice.

To summarize, we make the following main contributions:

1. We propose *k-subscription*: the first obfuscation-based approach to hide a user's interests in microblogging services. Our approach encourages users to follow $k-1$ noise channels apart from each channel they want, so as to hide (i) their real interests in a set of k channels, and (ii) other users' interests in the microblogging service.
2. To quantify the effectiveness of our approach, we introduce a new notion: the *Disclosure Probability* P_C. This is the service's confidence that a user is interested in channel C.
3. We present an analytic evaluation of our approach and derive closed-form formulas for the disclosure probability. These formulas suggest that the disclosure probability can be made predictably small by fine-tuning the obfuscation level k.
4. We evaluate *k-subscription* in a more realistic scenario using simulations, which are based on models derived from a real-world dataset with sensitive channels from Twitter.
5. We implemented our system as a plug-in for the Chrome browser using Twitter as case study. We experimentally evaluate our prototype and show that it has minimal bandwidth requirements and negligible latency to browsing experience.

2. SYSTEM DESIGN

2.1 Threat Model

We assume the existence of a microblogging service where users are able to follow individual channels. A channel can be the account of a physical person, of an entity such as a corporation, of a news site, of a politician's office, and so on. Additionally, we assume that the microblogging service is capable of recording the users' interests by observing which channels each user follows. The information about the users' interests, which is property of the microblogging service, could be later sold to advertisers [21], and could be used for a variety of purposes, all of which are beyond the control of individual users [12]. We view this capability of the microblogging service as a potential concern for the users' privacy, and we would like to develop mechanisms that hide the users' real interests from the microblogging service.

In this work we assume an "honest but curious" microblogging service. In this aspect, the microblogging service may try to find the user's interest based on the channels the user is following, but it will not try to "cheat" by actively interfering with the process users are employing to protect their privacy, or try to gain more information than what a user is willing, or required, to give. For example, the microblogging service will not create fake channels or fake users in order to break the anonymity of ordinary users. We think that this "honest but curious" model is reasonable in practice, as popular microblogging services have a reputation they do not want to jeopardize by becoming hostile against their own users. Therefore, we expect such microblogging services to only try to passively gain knowledge based on data given by their users.

We also assume that when users *follow* channels, they act as consumers of information and refrain from interacting with any channel by posting information, replying, retweeting, or sharing their interests in any other way. Indeed, if a user starts posting about a sensitive issue, it will be easy for the microblogging service to identify the user's interests. We believe that most users want to find and consume information about a sensitive issue, and they will not take the risk of being identified by posting information about it. If some users would like to post, reply, or retweet anonymously about a sensitive issue, they may use *k-subscription* in combination with alternative solutions, such as #h00t [3] and Hummingbird [5].

Notation	Explanation
S	: Set of sensitive channels that can be followed
C	: Sensitive channel
U	: Number of all users in the system
U_C	: Number of users actually interested in channel C
U_{R_C}	: Number of users following channel C at random
p_C	: Popularity of channel C ($p_C = U_C/U$)
P_C	: Probability that a user following C is interested in C
N	: Number of sensitive channels a user is interested in
k	: Obfuscation level (per channel)

Table 1: Summary of Notation

2.2 Our Approach: k-subscription

Table 1 summarizes the notation we use throughout this section. Assume that user A is interested in following channel C, which deals with a sensitive issue, such as a medical condition. If user A follows only this channel, the microblogging service would easily figure out that A is interested in this medical issue. In this paper we propose *k-subscription*: a system that makes sure that the microblogging service is not able to pinpoint A's interests with reasonable accuracy. To do so, *k-subscription* follows an obfuscation-based approach, which advocates that along with each channel C the user is interested to follow, she should also follow $k-1$ other channels (called "noise" channels). The number of noise channels are such that the microblogging service will not be able to determine A's interest with high probability, and will not be able to identify the actual set of users interested in each specific channel. All the noise channels are randomly chosen from a set S of "sensitive" channels. Note that A's real interests are also members of S.

2.3 Uniform Sampling

When a user wants to follow channel C, *k-subscription* encourages the user to follow $k-1$ other channels as well (say C_1, C_2, ..., C_{k-1}). In this way, the microblogging service will not know whether the user is actually interested in channel C or one of the C_1, C_2, ..., C_{k-1}. In our first algorithm, *k-subscription*-UNIF, these channels are chosen randomly with uniform probability from S. Algorithm 1 presents the pseudocode for *k-subscription*-UNIF.

Algorithm 1 *k-subscription*-UNIF: Choose noise channels uniformly from the set S

```
F = ∅; // initialize the set of channels to follow
for (i = 1 ; i ≤ k − 1 ; i + +) do
    C_i = randomly select a channel from set S ;
    S = S \ C_i ; // remove C_i from S
    F = F ∪ C_i ; // add C_i in the set of channels to follow
end for
F = F ∪ C ; // add C in the set of channels to follow
Follow all Channels in F in a random order ;
```

k-subscription-UNIF is a naive but powerful approach for obfuscation and we use it as a basic principle for our method. However, this approach leads to some practical problems. In case that not all channels enjoy the same popularity, then uniformly sampling from S may result in higher disclosure probability for the more popular channels. Thus, we discuss an improved version in the next section.

2.4 Proportional Sampling

A user following a popular channel (say C) along with several unpopular ones has a higher probability of being interested in C than in the rest of them. Capitalizing on this knowledge, the microblogging service has a better chance of finding those users who follow popular channels. To mitigate this issue, we propose *k-subscription*-PROP that sample channels from set S according to their popularity. Assume that U_C is the number of followers of channel C and $U_S = \sum_{\forall C \in S} U_C$ is the number of followers of all channels in S. Thus, instead of sampling all channels with probability $1/|S|$, we sample channel C with probability U_C/U_S. In Twitter, the popularity of a channel can be inferred by the number of users following the respective account. In other microblogging services similar metrics are available to determine the popularity of a channel. *k-subscription* with proportional sampling for adding noise does not affect the respective channel popularity.

2.5 Following Multiple Channels

Users may be interested in following more than one sensitive channels. Using *k-subscription*, users just need to select $k-1$ other noise channels to follow for *each* channel C they are interested in. Therefore, a user interested in following N channels will result in following $k \times N$ channels in total. However, it is very likely that a user will be interested in N sensitive channels that are *semantically related*. This case may significantly increase the disclosure probability. Indeed, the microblogging service can easily find the correlated channels: it will get all the channels a user is following, classify them into semantic categories, and identify the sets of channels that are semantically related. If there is only one set of related channels, it is more probable that the user actually follows them, and the remaining unrelated channels are the selected noise.

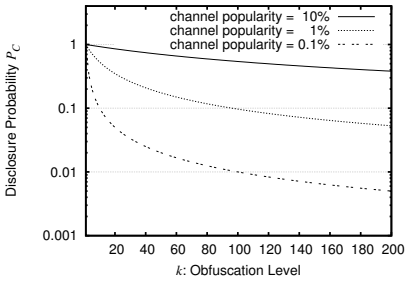
One way to address this issue could be the following: whenever a user is interested in N related channels, the $(k-1) \times N$ noise channels could be selected in N-tuple groups, so that each N-tuple consists of N related noise channels. However, this approach has a certain limitation: the microblogging service and *k-subscription* may use different similarity metrics to identify related channels. For instance, the microblogging service may use a more fine-grained similarity metric to find out the actual related channels.

Fortunately, *k-subscription* is able to protect users' interests even when they are interested in multiple semantically related channels. Although a user will actually follow the set of N related sensitive channels she is interested in, which can be identified by the microblogging service, there will be a significant number of other users that also follow the same set of N related channels due to random noise channel selections, i.e., without being interested in them. This is due to the increased random selections when users are interested in multiple channels. Thus, the microblogging service will not be able to know which of the users following all these N related channels are actually interested in them.

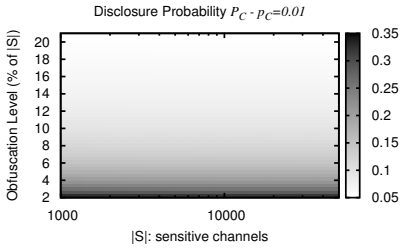
3. ANALYTICAL EVALUATION

3.1 Analysis of k-subscription-UNIF

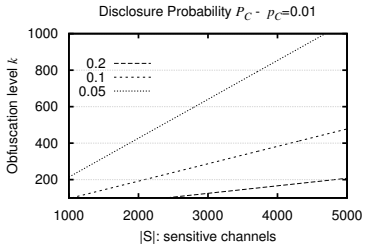
The disclosure probability P_C is the probability (as it can be calculated by the microblogging service) that a user who follows channel C is really interested in C. In our analysis we assume that the microblogging service is able to infer each channel's popularity U_C by the number of its followers or other external information, the number of users U that have adopted *k-subscription*, the size of set S that is publicly released, and the value of k used by each user, e.g., as a user subscribes to these k channels in a short period. Since along with channel C a user follows $k-1$ other channels as well, the disclosure probability is $P_C = 1/k$. However, for large values of k (i.e., in cases where the user wants to add a lot of noise) the microblogging service has a more effective way to increase its certainty about the interest of a user in a particular channel C. It knows that the U_C users who are interested in channel C actually follow it. At the same time, however, there are $U - U_C$ other users



(a) P_C as a function of k for different channel popularities.



(b) P_C as a function of $|S|$ and k, shown as a percentage of $|S|$, for channel popularity 1%.



(c) P_C as a function of $|S|$ and k, for channel popularity 1%.

Figure 1: Disclosure Probability P_C of *k-subscription*-UNIF as a function of the obfuscation level k and the size of S.

that are *not* interested in C, who may have randomly included C among their noise channels. The probability of C being included in the set of channels followed randomly by a user interested in a channel *different* than C is bounded by $1-(1-1/|S|)^{k-1}$, as this user will select $k-1$ channels randomly from S, which also includes C. Therefore, the average number of the $U-U_C$ users not interested in C that will follow C randomly as noise (U_{R_C}) are less than $(U-U_C)\times(1-(1-1/|S|)^{k-1})$. So, the ratio of users following C who are really interested in C is less than

$$\frac{U_C}{U_C+(U-U_C)\times(1-(1-1/|S|)^{k-1})}$$

Since the microblogging service does not know who are the users U_C interested in the channel C, it can only assume that all users following C are interested in C. The probability of a user following C actually being interested in C (denoted as P_C) is given by:

$$\begin{aligned} P_C &< \max(1/k, \frac{U_C}{U_C+(U-U_C)\times(1-(1-1/|S|)^{k-1})}) \Rightarrow \\ P_C &< \max(1/k, \frac{p_C}{p_C+(1-p_C)\times(1-(1-1/|S|)^{k-1})}) \end{aligned} \quad (1)$$

where p_C is the channel's popularity. We see that the total number of users U and the number of users U_C interested in channel C do not affect the disclosure probability. Instead, the channel's popularity p_C, the parameter k, and the total number of channels $|S|$, are the key factors that affect the disclosure probability.

3.1.1 Estimating the Disclosure Probability

First we arbitrarily fix $|S|$ to 1,000 channels. Figure 1(a) shows how the disclosure probability P_C changes as a function of k (the level of obfuscation). We see that when the popularity p_C of a channel is rather high (i.e., 10%), then it is difficult to obfuscate it with the *k-subscription*-UNIF approach. Indeed, when as many as 10% of the users are interested in channel C, then it would take a significant percentage of the rest 90% to include channel C among their noise channels, which is very difficult to achieve. However, when popularity is around 1%, then it is much easier to obfuscate it using this approach. Indeed, for $k=100$ the disclosure probability is as low as 0.1, which means that the microblogging service can not state with confidence larger than 10% that a user A who follows channel C is really interested in C. Fortunately, when the popularity of C is even lower (i.e., around 0.1%), the disclosure probability becomes 0.01 for obfuscation levels as low as 100. That is, the microblogging service can not say with confidence higher than 0.01 that a user who follows C is really interested in C.

In our next figure we explore how the disclosure probability changes as a function of the number of sensitive channels $|S|$ and the obfuscation level k. That is, if we double $|S|$ how should we increase k so as to have the same disclosure probability? Figure 1(b) shows a plot of the P_C as a function of $|S|$ and k. Note that the obfuscation level k in the y-axis is shown *not* as an absolute number but as a percentage of the number of sensitive channels. From this figure we clearly see that lines of the same color run horizontally. Horizontal lines mean that the value is the same for constant y (obfuscation level as a percentage of $|S|$). This implies that as long as the obfuscation level is a constant percentage of the size of the set of sensitive channels $|S|$, the disclosure probability remains constant. To put it simply, if we double the number of sensitive channels, we need to double the obfuscation level k (in absolute numbers) in order to keep the disclosure probability constant.

To explore the relation between $|S|$ and k even further, Figure 1(c) shows the iso-probability contours of the disclosure probability P_C as a function of the number of sensitive channels (i.e., $|S|$ on x axis) and the obfuscation level (i.e., k on the y axis). We plot the contours for probabilities 0.05, 0.1, and 0.2. Interestingly, we see that the iso-probability contours appear as straight lines, clearly implying an almost linear relation between $|S|$ and k. That is, doubling $|S|$ would require a twice as high k in order to keep the probability of disclosure to the same level. Similarly, if we can afford to double the obfuscation level, we can provide the same disclosure probability for twice as many sensitive channels. Or, equivalently, if we are forced to half the obfuscation level, we can still provide the same disclosure probability but only for half as many sensitive channels. These results are very encouraging in the sense that we can still achieve the levels of P_C we are comfortable with, even when we are forced to use small obfuscation levels.

Figure 2 shows the impact that the number of sensitive channels $|S|$ and the channel popularity p_C have on the disclosure probability. The obfuscation level k is set to 100. We see that the iso-probability contours are almost straight anti-diagonal lines indicating an almost inversely proportional relation between $|S|$ and p_C.

3.1.2 Finding a Reasonable Size for S

We now analyze how large S should be and how we can influence it. One can easily see that the larger S is, the higher the disclosure probability will be (for a constant obfuscation level k). Therefore, we do not want the set S to be very large. On the other hand, a very small S would easily give away a user's true interests, and limit the users' choice for sensitive channels. Indeed, if S contains two channels, say C_1 and C_2, if a user follows C_1 (no matter whether she follows C_2 or not), the microblogging service will be able to conclude with probability 1/2 that the user is interested in

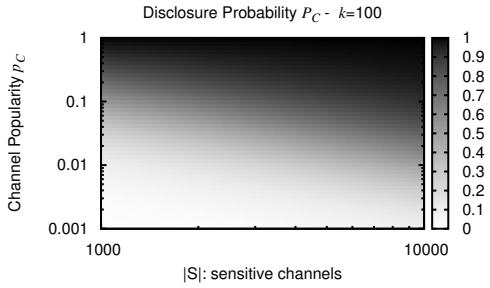


Figure 2: Disclosure Probability P_C of *k-subscription*-UNIF as a function of the size of S and channel popularity p_C. The obfuscation level k is set to 100.

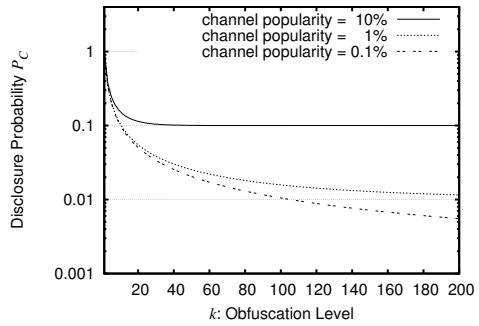


Figure 3: Disclosure Probability P_C of *k-subscription*-PROP as a function of the obfuscation level k. Sampling is proportional according to a channel's popularity.

C_1. In the same spirit, if S contains n members, the microblogging service will be able to conclude with probability at least $1/n$ that the user is interested in the channel she follows. As a result, S should be large enough so that the probability $1/n$ is small enough, so as not to be useful for the microblogging service. In the absence of any other information, the microblogging service is able to conclude with probability U_C/U that a random user is interested in channel C. Therefore, the set S should be large enough so that $1/|S| < U_C/U$. We can estimate U_C from external sources, or based on the number of followers of channel C.

3.2 Analysis of k-subscription-PROP

Let us now derive closed formulas for *k-subscription*-PROP. The probability of C being included in the set of channels followed randomly by a user interested in a *different* channel than C is bounded by $1-(1-U_C/U)^{k-1}$, as this user will choose at random $k-1$ channels from S, and the probability of choosing C at each individual choice is U_C/U. The average number of users not interested in C who will follow C randomly as noise (U_{R_C}) are less than $(U-U_C)\times(1-(1-(U_C/U))^{k-1})$. So, the ratio of users who follow C and are interested in C is less than

$$\frac{U_C}{U_C+(U-U_C)\times(1-(1-U_C/U)^{k-1})}$$

Since the microblogging service does not know who are the U_C users interested in C, it can assume that all users following C are interested in C. The probability of a user following C being interested in C (i.e., the disclosure probability) is bounded as follows:

$$\begin{aligned} P_C &< \max(1/k, \frac{U_C}{U_C+(U-U_C)\times(1-(1-U_C/U)^{k-1})}) \Rightarrow \\ P_C &> \max(1/k, \frac{p_C}{p_C+(1-p_C)\times(1-(1-p_C)^{k-1})}) \end{aligned} \quad (2)$$

where p_C is the channel's popularity. We observe that instead of the total number of users U and the number of users U_C that follow the channel C, the disclosure probability is affected only by channel's popularity p_C, number of channels S, and obfuscation level k.

Figure 3 shows how disclosure probability changes with the level of obfuscation k. We see that our *k-subscription*-PROP approach is able to efficiently hide popular channels. Indeed, for a popularity of about 10%, it is able to reach a disclosure probability of 0.1 using only $k=40$. When the popularity is 1%, even small obfuscation levels such as $k=50$ can lead to disclosure probability as low as 0.02, which is very encouraging. One can notice in Figure 3 that as k increases for 10% popularity, the disclosure probability tends to flatten out and in no case drops below 0.1. The reason is that for a channel with 10% popularity, the disclosure probability can never fall below 10% no matter how large the obfuscation level we use is. There is a simple explanation for this: without taking any channel-following information into account, the microblogging service knows that 10% of the population is interested in channel C. Hence, the microblogging service can safely assume that a user is interested in C with probability 0.1.

3.3 Analysis for Multiple Channels

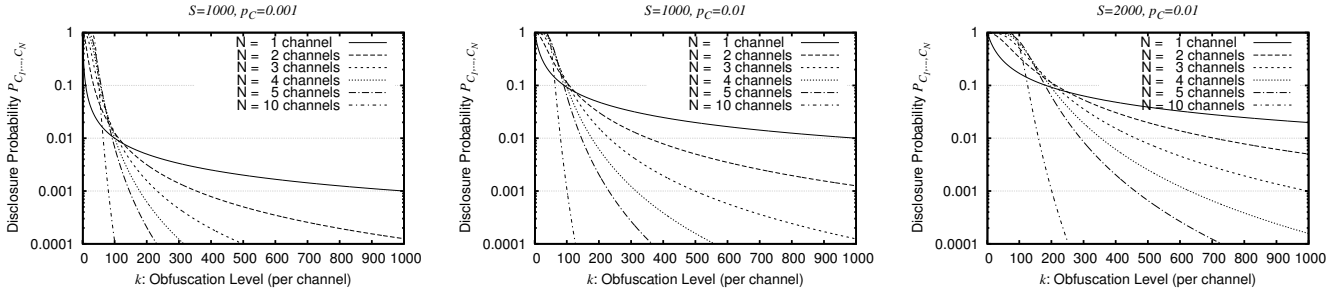
A user may want to follow more than one sensitive channels, which may be semantically related to each other. For simplicity, we assume that all users U are interested in exactly N sensitive channels from S. Each user that is actually interested to follow the N channels $C_1,...,C_N$ will also follow $(k-1)\times N$ noise channels based on random choices from S. Besides the $U_{C_1,...,C_N}$ users that are interested in these N channels, there will be some other users that will follow the same set of N channels without being interested in all of them, due to random noise channel selections. These users contribute to hide the actual interests of the $U_{C_1,...,C_N}$ users.

Since the microblogging service does not know the users who are actually interested in channels $C_1,...,C_N$, it can only assume that all users following these channels are interested in them. The disclosure probability $P_{C_1,...,C_N}$ that a user following all these N channels is actually interested in them is equal to:

$$\frac{U_{C_1,...,C_N}}{U_{C_1,...,C_N}+(U-U_{C_1,...,C_N})\times\binom{|S|-N}{(k-1)N-N}/\binom{|S|}{(k-1)N}} = \frac{p_{C_1,...,C_N}}{p_{C_1,...,C_N}+(1-p_{C_1,...,C_N})\times\binom{|S|-N}{(k-1)N-N}/\binom{|S|}{(k-1)N}} \quad (3)$$

where $\binom{|S|-N}{(k-1)N-N}/\binom{|S|}{(k-1)N}$ is the probability that a user selects randomly these N channels from the set S with $(k-1)\times N$ random choices when using the *k-subscription*-UNIF approach. We estimate this probability with a hypergeometric distribution, where all the successes N in the population S should be drawn with $(k-1)\times N$ attempts. Since we assume that all channels have the same popularity, the *k-subscription*-PROP approach has exactly the same behavior with *k-subscription*-UNIF in this analysis. $p_{C_1,...,C_N}$ is the popularity of the N-tuple of sensitive channels, i.e., the percentage of users actually interested in all these $C_1,...,C_N$ channels.

We want to explore how the disclosure probability changes with the number of channels N that a user may be interested in. We assume that the users interested in N channels are $U_{C_1,...,C_N}=U_C/N$, i.e., they are reduced by N times. Note that we assume a hyperbolic decrease of the users as N increases, instead of an



(a) $|S|$=1,000, channel popularity=0.1% (b) $|S|$=1,000, channel popularity=1% (c) $|S|$=2,000, channel popularity=1%

Figure 4: Disclosure Probability as a function of the obfuscation level k when users are interested in one up to ten sensitive channels, for different size of S and channel popularity p_C.

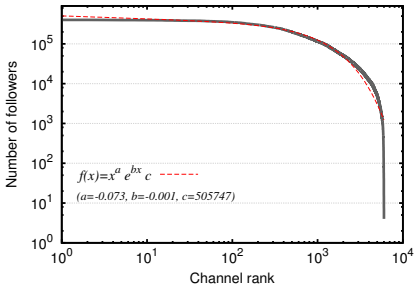


Figure 5: Distribution of the sensitive channels popularity.

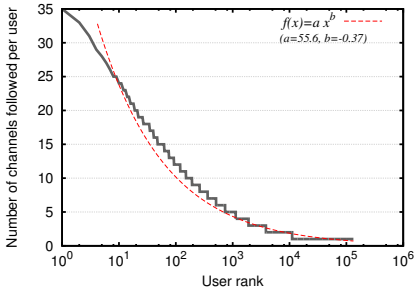


Figure 6: Distribution of the number of sensitive channels followed by a user.

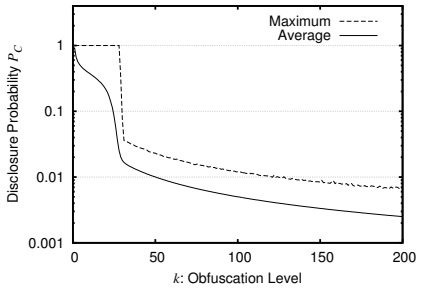


Figure 7: Disclosure probability as a function of k using realistic simulations.

exponential decrease, because we believe that these N channels will be probably semantically related. We set the size of S to $1,000$ and $2,000$ sensitive channels and we assume that all channels have the same popularity p_C, which is set to 0.1% and 1%.

Figure 4 shows the disclosure probability as a function of k for different values of N, ranging from 1 up to 10. We see that as the number of channels N increases, the disclosure probability is increased for low values of k, but it decreases significantly for higher k. The increase for low k values is because the users interested in N channels are reduced just by N times, following a hyperbolic growth, while the probability of randomly selecting these N channels for the rest users is reduced significantly by following a hypergeometric distribution. Thus, it is unlikely for the rest users to follow all these N channels at random with low k values. This means that for users interested in many sensitive channels we need to use a higher k to achieve a low disclosure probability.

In contrast, for higher values of k, we see a significant reduction of the disclosure probability when users are interested in more channels. This is because the users interested in N sensitive channels follow $k \times N$ channels in total, so we have more random selections for higher N values. For instance, when $N = 5$ and $k = 200$, each user follows $1,000$ channels from S, i.e., all the existing channels in S when $|S| = 1,000$. The same happens in case of $N = 10$ and $k = 100$. When the size of S and channel popularity p_C increase, the disclosure probability increases respectively, according to Equation 3. However, a proper selection of a higher k value results in a much lower disclosure probability, as the users' interests can be efficiently hidden among the random selections of other users. Our experimental evaluation in Section 6 shows that the network bandwidth and latency when following few hundreds of noise channels are negligible, so our approach is able to protect the users' privacy even when they are interested in many sensitive channels that are probably semantically related.

4. SIMULATION-BASED EVALUATION

To evaluate *k-subscription* in a more realistic setup, where users are interested in a different number of sensitive channels, and sensitive channels have different popularities, we built a realistic Monte Carlo simulator. The simulator assigns a random popularity p_C to each channel following a similar distribution to real-world sensitive channels. First, each user randomly selects the number of channels N that she is interested in following, based on a distribution similar to real-world users' selections. We assume that all N channels are semantically correlated. Then, the user selects these channels one-by-one at random, proportionally to channel's popularity. The noise selection is performed with *k-subscription*-PROP.

To simulate a realistic popularity distribution of sensitive channels, we selected a set S of $7,000$ sensitive channels using Twellow [1], a website that categorizes Twitter accounts according to their subject. The selected channels correspond to Twitter accounts dealing with health, political, religious, and other sensitive issues. We estimate the popularity of each channel based on its number of followers, i.e., U_C. Figure 5 shows the distribution of sensitive channel popularity in our dataset. We see that this distribution can be approximated very well using a power law with exponential cut-off model. We use this approximation in the simulator to assign a popularity p_C in each channel. We also see that only a small percentage of the sensitive channels exhibit relatively high popularity, which increases the disclosure probability. In contrast, the majority of sensitive channels have low popularity, which results in low disclosure probability even for low values of k.

To simulate a realistic distribution of the number of sensitive channels N that each user is interested in, we used the same real-word dataset of sensitive channels. From the total $7,000$ channels, we used the Twitter API to collect the user IDs of the followers of 500 sensitive channels related to disability issues, and we measured

the number of occurrences of each user ID. This is the number of channels belonging in S that each user in our dataset follows. In this analysis we found more than $530,000$ unique users. Figure 6 shows the respective distribution, which is approximated with a typical power law function. This approximation is used in our simulations for realistic user selections. Also, we observe that only 0.85% of the users follow more than 4 sensitive channels, while 91.65% of the users follow just one sensitive channel.

The simulator keeps two counters per each channel: the number of users that (i) select this channel as actual interest (U_C), and (ii) select this channel as noise (U_{R_C}). Before exiting, the simulator reports the disclosure probability per channel, which is $max(1/k, U_C/(U_C + U_{R_C})$. Additionally, it keeps two lists per user: (i) the channels she is interested in (C_i), and (ii) the channels she selects as noise (C_n). This way, the simulator reports the disclosure probability per user, based on the set of sensitive channels the user is interested in (C_i). This probability is equal to $max(1/k, U_{C_i}/(U_{C_i} + U_{R_{C_i}}))$, where U_{C_i} the number of users interested in C_i and $U_{R_{C_i}}$ the number of users that C_i is included within their C_n. This is because we assume that all channels in C_i are semantically correlated. Among all the disclosure probabilities reported per each user and each channel, the simulator reports the overall average and maximum disclosure probability. We repeat each simulation for 100 times and we use the average values.

We set $|S|$ to 1,000 channels, U to 1,000,000 users, and we vary k from 1 to 200. Figure 7 shows the average and maximum disclosure probability reported by the simulator as a function of k. We see that *k-subscription* achieves a low average disclosure probability over all channels and users, which decreases rapidly with k. However, we see that the maximum disclosure probability found for an individual user remains equal to 1 for low values of k up to $k = 30$. This is because there is at least one user interested in an N-tuple that no other user has selected among her random noise choices, especially for large values of N. However, as k increases, we see that an increasing number of users tend to select a significant percentage of the channels in S as random choices, e.g., users with large value of N. As these users follow most of the channels in S, they tend to hide the actual users' interests, even for large and rare N-tuples, reducing effectively the maximum disclosure probability.

5. IMPLEMENTATION

To evaluate the feasibility and efficiency of *k-subscription* we have implemented a Twitter extension for the popular Chrome web browser. The extension uses Twitter API v1.1 and complies with the REST API Rate Limit. It is developed using Javascript and JQuery, Json2, OAuth and SHA-1 libraries.

Figure 8 shows the overall operation of *k-subscription* extension. Upon installation, users can follow Twitter accounts in exactly the same way, though Twitter's web interface or "Follow me" buttons in third-party web pages. To enhance user's privacy, *k-subscription* intercepts all follow requests and checks whether they correspond to sensitive channels contained in S. If so, the extension transparently subscribes the user to $k-1$ additional "noise" channels from S according to the *k-subscription*-PROP algorithm, where k can be configured by the user. These channels remain hidden and the user never interacts with them, providing exactly the same Twitter browsing experience as before. For this reason, the extension keeps a list of all "noise" channels and dynamically filters out the unsolicited tweets of these channels from user's feed. Other affected information, such as the number of channels followed, is adjusted appropriately by excluding the effect of the "noise" channels.

At the first run, the extension downloads the set S of sensitive channels used for obscuring user's selections. The set includes information about each channel and its number of followers to improve "noise" selection. The user can interfere with "noise" selection by proposing channels with predefined features such as language and country. Users can disable the effect of *k-subscription* on a follow request if they consider the related channel as non-sensitive. When a user unfollows a sensitive channel, the extension transparently removes its corresponding "noise" channels as well.

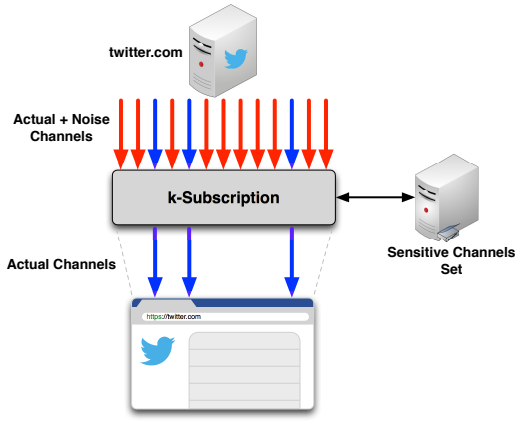


Figure 8: Overall operation of the *k-subscription* browser extension for Twitter. Whenever a user follows a new sensitive channel, *k-subscription* transparently follows additional "noise" channels and removes the "noise" from user's feed.

We envision that the set of sensitive channels S along with the project in general would be maintained by the broader community of users and/or Non-Governmental Organizations (NGO) that have a specific view towards protecting privacy. Hence, S can be seeded by an initial set of sensitive channels and further improved through human intervention and participation of the community. Similar privacy-concerned projects, such as Tor, enlist the help of volunteers to maintain and improve its networks of routers. We expect that similar approaches can be applied for *k-subscription*.

Although *k-subscription* is effective at hiding the channels a user is interested in, a microblogging service may be able to find the user's real preferences by collaborating with URL shortening services. Users who click on a short URL are initially directed to the URL shortening service (which may be operated by the microblogging service, such as `t.co` in the case of Twitter) and then they are redirected to the actual URL. By monitoring which short URLs a user clicks, the microblogging service can learn the user's interests. We solve this problem within the *k-subscription* browser extension: whenever a user clicks on a short URL, *k-subscription* opens all short URLs posted in every channel the user follows. These URLs are resolved to the final destination URLs without the browser receiving a single byte from the targeted web pages. Then, the extension serves to the user only the actually requested URL. This way, the microblogging service will not be able to see which URLs a user clicks, as *k-subscription* transparently opens all of them.

6. EXPERIMENTAL EVALUATION

6.1 Environment and Dataset

For all our experiments we used a PC equipped with an Intel Core 2 Duo Processor E8400 with 6MB L2 cache, 4GB RAM, and an Intel 82567 1GbE network interface. To populate the set S with sensitive channels we used Twellow [1], a web site that categorizes Twitter accounts according to their subject. We created a set S with $7,000$ Twitter accounts dealing with health issues, political beliefs, abuses, religious preferences and more.

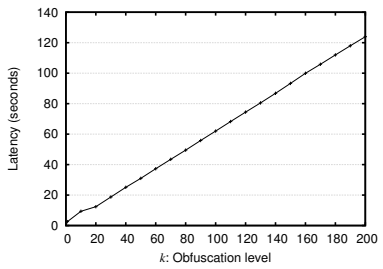


Figure 9: Time to follow a sensitive channel as a function of k.

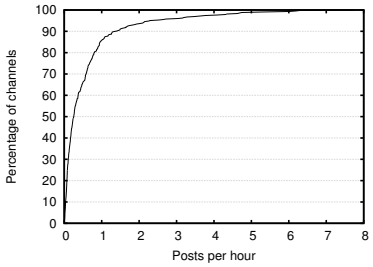


Figure 10: Number of tweets posted per channel per hour.

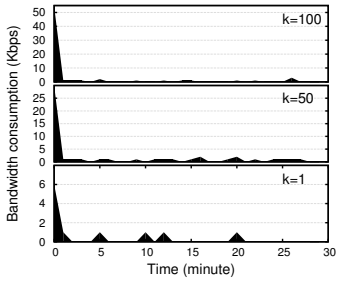


Figure 11: Bandwidth consumed for a user receiving tweets as a function of time.

6.2 Adding Channels

In this first set of experiments we set out to explore the delay imposed by *k-subscription* when adding several noise channels in order to follow and hide a sensitive channel the user is interested in. Figure 9 shows how much time it takes to follow a sensitive channel with *k-subscription* as a function of k, i.e., when also following $k-1$ noise channels. We repeated our measurement for each k 100 times with random choices, and we report the average values. We see that the latency is an almost linear function of k, as expected. Fortunately, the time to follow several tens of channels is not significant. Indeed, it takes a little more than 1 minute to follow 100 channels and around 2 minutes to follow 200 channels. Since this operation is done only once, i.e., when a user wants to follow a sensitive channel, we believe that it does not add any significant overhead. Moreover, this operation runs as a background process, so it does not affect the user's experience.

6.3 How Much Does the Noise Cost?

In our next experiment we tried to quantify how much more traffic is generated by the noise channels. To do so, we measured the total number of tweets generated by all channels, divided by each channel's lifetime and found the average number of tweets per channel per unit of time. The CDF of this function is shown in Figure 10. We see that the median channel (y=50%) generates less than one tweet (actually 0.25 tweets) per hour while 93% of the channels generate less than two tweets per hour. Overall, we see that the extra traffic generated by the noise channels should be very small. Even adding 100 noise channels generates no more than 25 tweets per hour, a negligible amount of traffic by most standards.

The reader will notice that the maximum posting rate that we have observed is about 6 posts per hour (averaged over the entire lifetime of the channel). Published statistics [24] suggest that the most prolific twitter accounts post as much as one tweet update per minute. Such accounts usually belong to news stations or even to automated programs (bots). Given that each tweet corresponds to just few hundred of bytes transferred over the network, even in such cases the resulting network overhead will be relatively low.

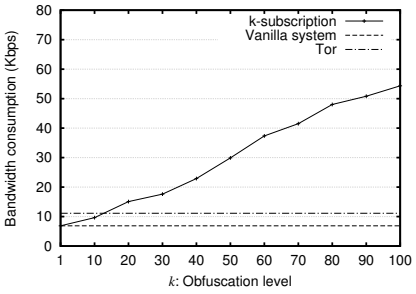
6.4 Bandwidth Consumption

When a user follows k channels for each subscription, she downloads roughly k times more information than she actually needs. However, we would like to see if the bandwidth needed for these downloads can be sustained by a home DSL Internet connection or not, and the respective network overhead in terms of used bandwidth. A user interested in N sensitive channels will receive tweets from $N \times k$ channels. The network overhead will be the same when a user is interested in one channel with N times higher k value. Thus, we evaluate our system while varying only the k parameter, assuming a user interested in a single channel. Figure 11 shows the traffic load generated by our implementation over a 30-minute period for a user following one sensitive channel with $k = 100$, $k = 50$, and $k = 1$ (i.e, without using *k-subscription*). We notice that the bandwidth consumption even in case of $k = 100$ is reasonably low, usually less than 1.5 Kbps. We see that even in case of the vanilla system (see $k = 1$) the bandwidth consumption is not significantly lower than in high values of k. In all cases it is usually between 0.5 and 1.0 Kbps. By manually inspecting the traffic we found that most of the bandwidth is used to download information like images, trends and recommendations, which does not depend on the value of k. The bandwidth used for downloading the actual tweets, which increase with the value of k, was found to be a small percentage of the total bandwidth consumption.

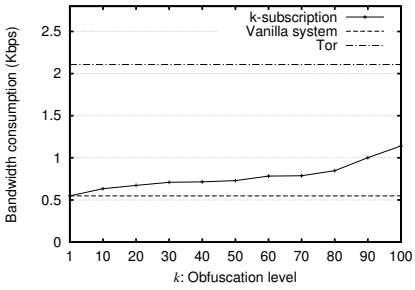
We observe a large spike at the beginning of each experiment, when we have just opened the browser and loaded the Twitter page. For instance, bandwidth consumption reaches 54 Kbps for $k = 100$, 29 Kbps for $k = 50$, and 7 Kbps for the vanilla system at the first second. During this initialization stage Twitter downloads all the necessary content (widgets, scripts, CSS, profile images, *etc.*). At this stage, *k-subscription* downloads lot of tweets from all k channels. As we discard most of the tweets (belonging to noise channels) and keep only tweets from channels that a user is interested in, we always download a larger chunk of tweets to be able to compound the user's actual timeline. For this reason we observe a relatively higher spike as k increases. To improve browsing latency, we transparently increase the page size to receive more tweets.

Note that profile images are cacheable, so *k-subscription* downloads the additional images (depending on k) only once, without affecting the overall bandwidth consumption. After the initial spike in the first few seconds, we see constantly low bandwidth consumption, which correspond to the low rate of incoming tweets. The average consumption in this 30-minute interval is 1.14 Kbps for $k = 100$, 0.71 Kbps for $k = 50$, and 0.54 Kbps for $k = 1$. Overall, we see that the total bandwidth consumed by *k-subscription* is not really an issue even if the user follows as many as 100 channels.

To evaluate the effect of the obfuscation level on bandwidth consumption while browsing Twitter with *k-subscription*, we plot in Figure 12 the bandwidth consumption as a function of k for two different stages: (i) when the user loads Twitter and downloads her timeline, which consists of the latest 20 tweets from the channels she is interested in, and (ii) when Twitter is idle and just receives new incoming tweets for 30 minutes. We see that the overhead is very low, even for large k, and can be easily handled by a home DSL or even a mobile connection. The bandwidth consumption is much lower in the idle stage, as expected, due to the low number of tweets per second, as shown in Figure 10. The increased bandwidth during initialization is because *k-subscription* asks for more tweets to display the default page of 20 tweets only from channels that user is interested in. However, the initialization lasts for just few seconds, e.g., 7.7 seconds for $k = 100$ and 2.8 seconds for the vanilla system. Thus, the increased bandwidth in Figure 12(a)



(a) Initialization stage (load 20 tweets)



(b) Idle stage (download incoming tweets)

Figure 12: Bandwidth consumption with *k-subscription*, Tor and vanilla system.

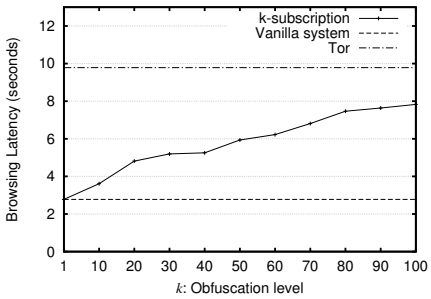


Figure 13: Browsing latency as a function of k when a user opens Twitter's main page.

corresponds to short term spikes, while the low bandwidth in idle stage (see Figure 12(b)) corresponds to much longer periods, as the user keeps Twitter's page open in the browser.

In Figure 12 we also compare the bandwidth consumption of *k-subscription* with a Tor browser. Although Tor offers a completely different type of anonymity than *k-subscription*, it could be used with a fake account as a different approach to hide user's interests. Thus, we evaluate *k-subscription* using the performance of Tor with a fake Twitter account as a baseline case. We see that Tor adds an additional bandwidth overhead due to its data encapsulation. In particular, the average packet size of Twitter traffic over Tor is 789 bytes, when the vanilla Twitter traffic has an average packet size of 239 bytes. This is the main reason that during the idle stage the bandwidth consumption of Tor is quite higher than the consumption of *k-subscription*, e.g., two times higher when $k = 90$. During the initialization stage, Tor has a higher bandwidth consumption than *k-subscription* with values of k up to 10, and lower consumption when k exceeds 10. This is due to the increased number of tweets downloaded at startup by *k-subscription* with high k values to construct a full page of useful tweets. However, as the initialization stage lasts only for few seconds, compared with idle stage, *k-subscription* adds less overhead in terms of bandwidth.

When *k-subscription* compounds the user's timeline, it continues to download tweets in the background until it reaches a certain number, which is constant for each k value. This way, *k-subscription* avoids leaking any information that Twitter could analyze to find out the channels a user is interested in.

6.5 Browsing Latency

In our next experiment we set out to explore the latency that *k-subscription* imposes to user's browsing experience. We instrumented our browser extension to measure the latency from the time that a user asks for one or more tweets till the time that the browser actually displays the relative information in the page, excluding any tweets from noise channels. This latency includes the time spent in network for downloading tweets, as well as the time spent in the CPU for excluding the noise and rendering the page. Note that the user's timeline is fully rendered when all the 20 tweets needed are received, despite the fact that more tweets are downloaded in the background to hide the actual user's interests.

Figure 13 shows the latency for displaying a page with *k-subscription* for several values of k when the user opens Twitter and loads her timeline. We see that the latency for downloading and displaying a full page with 20 tweets slightly increases with the number of noise channels, reaching to 7.7 seconds for $k = 100$ when without *k-subscription* (see $k = 1$) the latency to display the same page is 2.8 seconds. Therefore, a slight delay of less than 5 seconds is not expected to significantly affect the user's browsing experience, while, at the same time, it enhances her privacy. Selecting a smaller number of noise channels results in even lower latency. Note that this small delay is only observed at the initialization stage, due to the increased number of tweets needed to construct the user's actual timeline. When the browser remains open (idle stage) we do not observe any noticeable delay to render the incoming tweets, even at very high values of k. If an incoming tweet belongs to a noise channel we just drop it, else it is immediately given to user with no further delay. Thus, our approach does not impose any significant overhead to the browsing latency.

In Figure 13 we also compare the browsing latency of *k-subscription* and Tor. We see that Tor requires a much higher latency to display Twitter's page, close to 10 seconds. This is due to the longer path from user to Twitter through the anonymization network.

During the previous experiments we measured the CPU load of the browser, using the Linux's `time` utility. The CPU load was negligible for all values of k, always less than 1%, even for $k = 100$. Thus, our *k-subscription* browser extension does not add any considerable CPU overhead to the system.

7. RELATED WORK

Anonymous communications. One way to hide a user's accesses on the web is to use an anonymization service [6, 20]. Although such services can effectively hide a user's IP address, they can not hide the user's identity if the user is logged into a microblogging service or if a subset of user's previous web accesses is known [19]. Recently, obfuscation was used to hide a user's digital tracks. Kido et al. [13] protect the user's location privacy by sending false position data together with the true information.

Search engine query obfuscation. Howe and Nissenbaum [11] proposed *TrackMeNot*, a system designed to hide a user's real interest from a search engine. For each real query submitted to the search engine, TrackMeNot also submits several other queries to confuse the search engine and introduce doubt for the user's real queries. GooPIR [7] proposes an approach that is robust against timing attacks. For each real query, GooPIR constructs $k - 1$ other queries and submits all k of them at the same time. This way, the search engine cannot construct a timing model on the user's real queries. Murugesan and Clifton [16] propose *Plausibly Deniable Search* (PDS). Similar to GooPIR, each real query is accompanied by $k - 1$ other noise queries. Each real query, however, is also brought into a *canonical from* to prevent identifiability based on typos and/or grammar/syntax of the queries [2, 17]. Ye et al. [25] propose noise injection for search privacy protection. They give a lower bound for the noise queries required for perfect privacy and provide the optimal protection given the number of noise queries.

Although the above systems are very effective at hiding *one* real query in a crowd of k queries, a determined adversary may be able to find a user's interests by studying successive sequences of queries [4]. Indeed, if a user consistently generates authentic

queries on a particular topic, but the $k-1$ "noise" queries added are on several different topics, then the adversary may easily find the user's real interests using clustering approaches. To protect against clustering attacks, PRAW [9] generates dummy queries on topics related to the topics the user is interested in.

Our work shares ideas with the above works on search engine query obfuscation. However, it has a fundamental difference: in the field of search engine query obfuscation it is possible for some queries, especially the rare ones, to be submitted by only one user. Therefore, it is easy for the search engine to identify the users who submit rare queries and thus, to accurately find their interests. On the contrary, in *k-subscription* we always make sure that each channel, even the rare ones, is followed by lot of users. To put it simply: it is not how many "noise" channels a user follows – it is how many other users follow her channels of interest.

Hummingbird. Cristofaro et al. proposed *Hummingbird*, a system to provide privacy in Twitter [5]. The system assumes that a user (Alice) is interested in following a particular hashtag, e.g., from the New York Times (NYT). Hummingbird makes sure that neither Twitter nor NYT learn that Alice is interested in this hashtag. To achieve this, information providers (such as NYT) encrypt their tweets and information consumers (such as Alice) are able to decrypt the tweets matching the hashtags they are interested in.

Although Hummingbird is effective at hiding the hashtags Alice is interested in, and seems related to our work, we see two main differences with our approach: (i) Hummingbird requires the *explicit* collaboration of the information provider (e.g., NYT) who should encrypt its tweets appropriately, and distribute keys so that Alice will be able to decrypt the tweets matching the hashtags she is interested in. In contrast, our system does not require any collaboration from the information providers: it is implemented on top of Twitter as it is today. (ii) Although a user in Hummingbird is able to hide the hashtag she is interested in, she cannot hide the fact that she follows a particular channel (such as NYT). Our system is able to help Alice hide the fact that she is interested in the particular channel by making sure that she follows several other channels, and other people include this channel among their noise channels.

k-anonymity. Our work is similar to the concept of k-anonymity, which suggests that data should be anonymized in a way that any person in a released dataset should be indistinguishable from at least $k-1$ other persons in the same dataset [22]. To achieve k-anonymity, data are generalized so that any information that can uniquely identify a person will always point to at least k persons [18]. k-anonymity is frequently used together with l-diversity, which makes sure that all the persons in the same k-anonymity group do not have a common sensitive property [14].

8. CONCLUSION

Although microblogging services enable users to have timely access to their information needs through a publish-subscribe model, this creates major privacy concerns. As users declare all channels they are interested in following, the microblogging service is able to gather all their interests, including possible privacy-sensitive domains. To remedy this situation, we propose *k-subscription*: an obfuscation-based approach that encourages the users to follow $k-1$ additional "noise" channels for each channel they are really interested in following. We present a detailed analysis of our approach and show that by fine-tuning the k parameter we are able to reduce the confidence that the microblogging service has in knowing which channels each user is really interesting in. We have developed a prototype implementation as an extension for the Chrome browser using Twitter as case study. Our experimental evaluation shows that users may easily follow hundreds of noise channels with minimal run-time overhead when they receive news they are interested in. We believe that as an ever-increasing number of users turn to microblogging services for their daily information needs, privacy concerns will continue to escalate, and solutions such as *k-subscription* will become increasingly more important.

Acknowledgements

We thank our shepherd Matt Fredrikson and the anonymous reviewers for their valuable feedback. This work was supported in part by the FP7 project SysSec and the FP7-PEOPLE-2009-IOF project MALCODE, funded by the European Commission under Grant Agreements No. 254116 and No. 257007, by the German Federal Ministry of Education and Research under grant 01BY1111 / MoBE, and by the NSF through Grant CNS-1318415.

9. REFERENCES

[1] Twellow Directory. `http://www.twellow.com/categories/`.

[2] S. Afroz, M. Brennan, and R. Greenstadt. Detecting Hoaxes, Frauds, and Deception in Writing Style Online. In *IEEE Symposium on Security and Privacy*, 2012.

[3] D. Bachrach, C. Nunu, D. Wallach, and M. Wright. #h00t: Censorship Resistant Microblogging. *arXiv preprint arXiv:1109.6874*, 2011.

[4] E. Balsa, C. Troncoso, and C. Diaz. OB-PWS: Obfuscation-Based Private Web Search. In *IEEE Symposium on Security and Privacy*, 2012.

[5] E. De Cristofaro, C. Soriente, G. Tsudik, and A. Williams. Hummingbird: Privacy at the time of Twitter. In *IEEE Symposium on Security and Privacy*, 2012.

[6] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*, 2004.

[7] J. Domingo-Ferrer, A. Solanas, and J. Castellà-Roca. h(k)-private information retrieval from privacy-uncooperative queryable databases. *Online Information Review*, 2009.

[8] P. Eckersley. How Unique is Your Web Browser? In *Privacy Enhancing Technologies (PET)*, 2010.

[9] Y. Elovici, C. Glezer, and B. Shapira. Enhancing Customer Privacy While Searching for Products and Services on the World Wide Web. *Internet Research*, 2005.

[10] Epistolary. Rob's Giant BonusCard Swap Meet. `http://epistolary.org/rob/bonuscard/`.

[11] D. Howe and H. Nissenbaum. TrackMeNot: Resisting Surveillance in Web Search. *Lessons from the Identity Trail: Anonymity, Privacy, and Identity in a Networked Society*, 2009.

[12] R. Jones, R. Kumar, B. Pang, and A. Tomkins. I Know What You Did Last Summer: Query Logs and User Privacy. In *ACM Conference on Information and Knowledge Management (CIKM)*, 2007.

[13] H. Kido, Y. Yanagisawa, and T. Satoh. An Anonymous Communication Technique Using Dummies for Location-Based Services. In *IEEE International Conference on Pervasive Services (ICPS)*, 2005.

[14] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2007.

[15] J. R. Mayer and J. C. Mitchell. Third-Party Web Tracking: Policy and Technology. In *IEEE Symposium on Security and Privacy*, 2012.

[16] M. Murugesan and C. Clifton. Providing Privacy through Plausibly Deniable Search. In *SIAM International Conference on Data Mining (SDM)*, 2009.

[17] A. Narayanan, H. Paskov, N. Z. Gong, J. Bethencourt, E. Stefanov, E. C. R. Shin, and D. Song. On the Feasibility of Internet-Scale Author Identification. In *IEEE Symposium on Security and Privacy*, 2012.

[18] H. Park and K. Shim. Approximate Algorithms for k-Anonymity. In *ACM SIGMOD International Conference on Management of Data*, 2007.

[19] S. Peddinti and N. Saxena. On the Effectiveness of Anonymizing Networks for Web Search Privacy. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2011.

[20] M. Reiter and A. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1998.

[21] RT. Privacy betrayed: Twitter sells multi-billion tweet archive. `http://rt.com/news/twitter-sells-tweet-archive-529/`.

[22] L. Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2002.

[23] J. Turner. Countermeasure Radar Chaff, 1970. US Patent 3,544,997.

[24] Twitaholic. Top 100 Twitterholics based on Updates. `http://twitaholic.com/top100/updates/`.

[25] S. Ye, F. Wu, R. Pandey, and H. Chen. Noise Injection for Search Privacy Protection. In *International Conference on Computational Science and Engineering (CSE)*, 2009.