Privacy-Preserving Distributed Collaborative Filtering

Antoine Boutet, Anne-Marie Kermarrec, Davide Frey, Rachid Guerraoui, Arnaud Jégou

Privacy-Preserving Distributed Collaborative Filtering

Antoine Boutet, Davide Frey, Arnaud Jegou, Anne-Marie Kermarrec, Rachid Guerraoui

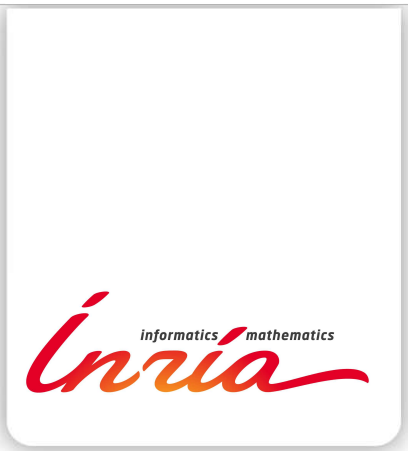
RESEARCH REPORT

N° 8253

February 2013

Project-Teams ASAP

ISSN 0249-6399 ISRN INRIA/RR--8253--FR+ENG



Privacy-Preserving Distributed Collaborative Filtering

Antoine Boutet [*], Davide Frey [†], Arnaud Jegou [‡], Anne-Marie Kermarrec [§], Rachid Guerraoui [¶]

Équipes-Projets ASAP

Rapport de recherche n° 8253 — February 2013 — 20 pages

Résumé : We propose a mechanism to preserve privacy while leveraging user profiles in distributed recommender systems. Our approach relies on (*i*) an original obfuscation mechanism hiding the exact profiles of users without significantly decreasing their utility, as well as (*ii*) a randomized dissemination algorithm ensuring differential privacy during the dissemination process. We evaluate our system against an alternative providing differential privacy both during profile construction and dissemination. Results show that our solution preserves accuracy without the need for users to reveal their preferences. Our approach is also flexible and more robust to censorship.

Mots-clés : privacy, recommendation systems, collaborative filtering, obfuscation

[*] antoine.boutet@inria.fr
[†] davide.frey@inria.fr
[‡] arnaud.jegou@inria.fr
[§] anne-marie.kermarrec@inria.fr
[¶] rachid.gerraoui@epfl.ch

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Système de filtrage collaboratif distribué respectant la vie privée

Abstract: Bien que la nature décentralisée des systèmes pair-à-pair supprime le problème des compagnies intrusives ayant un accès total aux données des utilisateurs, les fuites d'informations à caractère privé peuvent venir d'autres utilisateurs si le schéma de filtrage nécessite la manipulation de données personnelles. La seconde contribution de cette thèse propose (*i*) un mécanisme d'offuscation cachant le profil exact des utilisateurs sans trop dégrader son utilité (pour l'identification des utilisateurs ayant des centres d'intérêt similaires), et (*ii*) propose un processus aléatoire de dissémination. Plus précisément, notre protocole d'offuscation construit un filtre qui identifie les informations les moins sensibles du profil d'intérêt de l'utilisateur. En contrôlant la taille de ce filtre, le designer du système peut faire varier la quantité d'information échangée entre les utilisateurs pour former leur voisinage. Notre dissémination à caractère aléatoire, de son côté, évite qu'un utilisateur puisse savoir avec certitude l'opinion exacte des utilisateurs qui lui envoient des news.

Key-words: vie privée, systèmes de recommandation, filtrage collaboratif, obfuscation

1 Introduction

Collaborative Filtering (CF) helps users manage the ever-growing volume of data they are exposed to on the Web [17, 10]. In its "user-based" form [22], CF consists in leveraging interest similarities between user profiles to recommend relevant content.

Clearly, such systems induce a trade-off between ensuring user privacy and enabling accurate recommendations. Decentralization removes the monopoly of a central entity that could commercially exploit user profiles. However, users in a decentralized setting may in this case directly access the profiles of other users. Preventing such local privacy breaches is the challenge we address in this paper. We do so in the context of a news item decentralized CF system.

We propose a twofold mechanism : (*i*) an obfuscation technique applied to user profiles, and (*ii*) a randomized dissemination protocol satisfying a strong notion of privacy. Each applies to one of the core component of a decentralized user-based CF system : the user-clustering and the dissemination protocols. User clustering builds an interest-based topology, implicitly connecting users with similar preferences : it computes the similarity between profiles that capture the opinions of users on the items they have been exposed to. The dissemination protocol propagates the items along the resulting topology.

Our obfuscation technique prevents user machines from exchanging their exact profiles while constructing the interest-based topology. The protocol computes similarities using coarse-grained obfuscated versions of user profiles that reveal only the least sensitive information. To achieve this, it associates each disseminated item with an *item profile*. This profile aggregates information from the profiles of the users that like an item along its dissemination path. This reflects the interests of the portion of the network the item has traversed, gathering the tastes of a community of users that have liked similar items. Our protocol uses this information to construct filters that identify the least sensitive parts of user profiles : those that are most popular among users with similar interests. By controlling the size of these filters, system designers can therefore tune the amount of sensitive information exchanged between users.

Albeit simple and lightweight, this *obfuscation* technique prevents any user from knowing with certainty the exact profile of another user. Moreover, it achieves this without significantly hampering the quality of dissemination : the obfuscated profile reveals enough information to connect users with similar interests.

Our dissemination protocol ensures differential privacy. Originally introduced in the context of statistical databases [11], differential privacy bounds the probability of the output of an algorithm to be sensitive to the presence of information about a given entity – the interests of a user in our context – in the input data. We apply differential privacy by introducing randomness in the dissemination of items. This prevents malicious users from guessing the interests of a user from the items it forwards.

We compared our protocols with a non-private baseline as well as with an alternative solution that applies differential privacy to the entire recommendation process. This alternative constitutes a contribution of this paper in its own right. Our extensive results show that our twofold mechanism provides a good trade-off between privacy and accuracy. For instance, by revealing only the least sensitive 30% of a user profile, and by randomizing dissemination with a probability of 0.3, our solution achieves an F1-Score (trade-off between precision and recall) of 0.58, against a value of 0.59 for a solution that discloses all profiles, and a value of 0.57 for the differentially private alternative in a similar setting. Similarly, malicious users can predict only 26% of the items in a user's profile with our solution, and as much as 70% when using the differentially private one. In addition, our mechanism is very resilient to censorship attacks, unlike the fully differentially private approach.

The rest of this paper is organized as follows, Section 2 presents our system model, Section 3

highlights the potential privacy breaches while Section 4 and Section 5 describe, respectively, our obfuscation mechanism and our randomized dissemination protocol. Section 6 presents our experimental set up and Section 7 presents our performance evaluation. Section 9 concludes the paper.

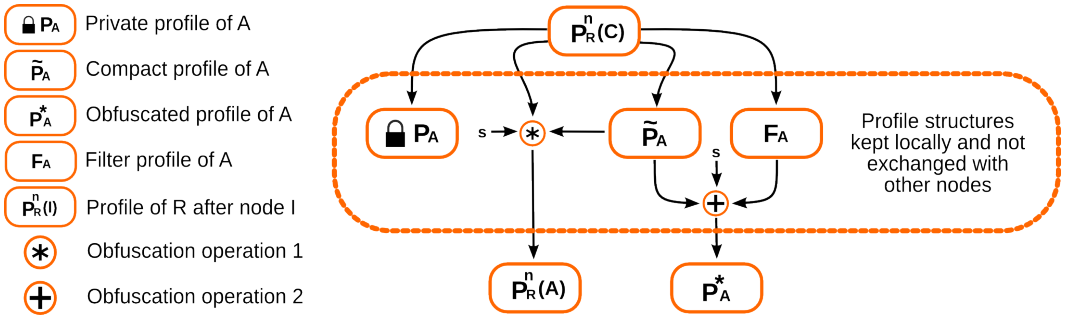


FIGURE 1: *Operations performed on profiles when node A likes a received item R from C.*

2 System Model

We consider a decentralized user-based collaborative filtering (CF) news item recommender. The axiom underlying user-based CF is that users who exhibited similar tastes in the past are likely to be interested in the same items in the future. When a user generates a new item, or expresses a positive opinion on a received item, the system spreads it to users with similar interests. In decentralized system, this process relies on two core components : *user clustering*, and *item dissemination*.

User clustering aims at identifying the k nearest neighbors of each user. To achieve this, it associates each user with a profile – a vector capturing her interests. It then computes similarities between the profiles of pairs of users. In the absence of global knowledge, a decentralized approach builds and maintains an interest-based overlay topology where each user is connected to users with similar interests. Epidemic protocols are particularly effective in building and maintaining such overlays [25, 6].

Dissemination also operates in a fully decentralized manner. In this paper, we consider a simple epidemic protocol : a user that generates an item, or that expresses a positive opinion on a received one, forwards it to her k interest-based neighbors.

2.1 Users Clustering

User clustering maintains a dynamic interest-based topology consisting of a directed graph $G(U, E)$, where vertices, $U = u_1, u_2, u_3, ...u_n$, correspond to users, and edges, $E = e_1, e_2, e_3, ...e_n$, connect users that have the most similar opinions about a set of items $I = i_1, i_2, ..., i_m$. The system is decentralized : each node records the interests of its associated user, u, in a *user profile*, a vector of tuples recording the opinions of the user on the items she has been exposed to. Each such tuple $P_u =< i, v, t >$ consists of an item identifier, i, a score value, v, and a timestamp, t, indicating when the opinion was recorded (the term 'node' refers the pair 'user/machine'). Profiles track the interests of users using a sliding window scheme : each node removes from its profile, all the tuples that are older than a specified *time window*. This allows the interest-based topology to

quickly react to emerging interests while quickly forgetting stale ones. In the following we focus on systems based on binary ratings : a user either *likes* or *dislikes* an item.

The interest-based topology is built by means of two gossip protocols running on each node. The lower-layer random-peer-sampling (RPS) [24] protocol ensures connectivity by maintaining a continuously changing random graph. The upper-layer clustering protocol [25] starts from this random graph and provides each node with its k closest neighbors.

Each node, n, in each protocol maintains a view, a data structure containing references to other nodes : each reference consists of (i) the node's IP address, (ii) its node ID, (iii) a profile, as well as (iv) a timestamp indicating when the associated profile was generated. Periodically, each protocol selects the entry in its view with the oldest timestamp and sends it a message containing its public profile with half of its view for the RPS and its entire view in case of the clustering protocol (standard parameters). In the RPS, the receiving node renews its view by keeping a random sample of the union of its own view and the received one. In the clustering protocol, instead, it computes the union of its own and the received view, and selects the nodes whose profiles are closest to its own according to a similarity metric. Several similarity metrics have been proposed [23], we use the Jaccard index in this paper.

2.2 Dissemination

A decentralized recommendation system exploits the above clustering scheme to determine which nodes should receive each item. Our model [1] achieves this through a gossip-based dissemination protocol. When a user receives an item she likes, the associated node assumes that this is an interesting item for other users with similar interests. It thus forwards the item to its neighbors in the interest-based topology. If, instead, the user marks an item as *dislike*, the node simply drops it.

3 Privacy Breaches in Distributed CF

The CF system as defined above does not integrate specific mechanisms to protect the privacy of users. While decentralization removes the prying eyes of *Big-Brother* companies, it leaves those of curious users who might want to discover the personal tastes of others. In this paper, we aim to protect a decentralized item recommender from malicious nodes that extract information in the two following ways : (i) from the profiles they exchange with other nodes ; and (ii) from the items they receive in the dissemination process.

As described in Section 2.1, nodes exchange profile information for maintaining the interest-based overlay. Profiles contains precise information about the interests of a user, which are potentially sensitive. Similarly, the dissemination protocol described in Section 2.2 is driven by the interest of users. As a consequence, a node a that receives an item from another node n can conclude that n *liked* that item. The predictive nature of this dissemination protocol thus also constitutes a leak of sensitive information.

The remainder of this paper presents a solution that addresses each of these two vulnerabilities by means of two dedicated components. The first is a profile obfuscation mechanism that prevents curious users from ascertaining the tastes of the user associated with an exchanged profile. The second, is a randomized dissemination protocol that hides the preferences of nodes during the item forwarding process.

4 Obfuscation Mechanism

Our first contribution is an obfuscation protocol that protects user profiles by (i) aggregating their interests with those of similar users, and (ii) revealing only the least sensitive information to other users. By tuning these two mechanisms, system designers can manage the trade-off between disclosed information and recommendation quality [18]. An excessively obfuscated profile that reveals very little information is difficult to compromise, but it also provides poor recommendation performance. Conversely, a highly accurate profile yields better recommendations, but does not protect privacy-sensitive information effectively. As we show in Section 7, our obfuscation mechanism provides good recommendation while protecting privacy.

4.1 Overview

Figure 1 provides an overview of our protocol. Each node maintains a profile that lists its opinions on the items it has been exposed to within the latest *time window*. We name it *private profile* because a node never shares its content with anyone else. In addition, we associate each item with an *item profile*, and each node, n, with three new profile structures : two private ones, the *compact profile* and the *filter profile* ; and the *obfuscated profile* potentially shared. The *item profile* aggregates the interests of users who liked an item. The *compact profile* provides a summary of node n's interests. The *filter profile* captures the main interest trends among the nodes that are similar to n. Finally, the *obfuscated profile* provides a filtered version of n's compact profile containing only the least senstitive information. Nodes do not strictly maintain the *compact*, *filter*, and *obfuscated profiles* : they are computed when needed, using the information from the *private* and *item* profiles.

4.2 Profile Updates

Private Profile A node updates its private profile whenever it expresses a like/dislike opinion about an item (line 3 and 8 in Algorithm 1), or when generating a new item. In either case, the node inserts a new tuple into its private profile. For liked items, this tuple contains the item identifier, its timestamp – indicating when the item was generated, a score value – 1 if the node liked the item, 0 otherwise, the item vector, and the item profile upon receipt. For a disliked item, the tuple contains only the item identifier, while the remaining fields are empty. Only the *private profile* gathers all the information on the interests of a user. Our obfuscation mechanism never reveals its contents to other nodes : it only uses it to update the other data structures as we discuss in the following.

Compact Profile Unlike private profiles, which contain item identifiers and their associated scores, compact profile stores liked items in the form of a d-dimensional bit array. This compact profile is produced using Random Indexing, an incremental dimension reduction technique [26, 15]. The basic idea of Random indexing is to reduce the private profile of users to a compact profile accumulating the random signatures, called *item vector*, of each liked item.

An item vector is generated by the source of an item and it consists of a sparse d-dimensional bit arrays with small number of randomly distributed 1's, with the rest of the elements set to 0. Several methods exist to build these vectors [7, 15]. We consider a vector size of d bits and a number $b << d$ of 1 values as system-wide parameters. The source of an item simply generates b distinct array positions and sets the corresponding bits to 1. It then attaches the resulting vector to the item before disseminating it. Finally, the compact profile of a user, u, is computed as the bitwise OR of the item vectors of the items u liked.

Algorithm 1: Receiving an item.

```
1  on receive (item < id^N, t^N >, item vector S^N, item profile P^N) do
2      if iLike(id^N) then
3          P ← < id^N, t^N, 1, S^N, P^N >
4          buildCompactProfile(S^N)
5          updateItemProfile(P^N)
6          forward(< id^N, t^N >, S^N, P^N)
7      else
8          P ← < id^N, t^N, 0 >
9  function buildCompactProfile()
10     for all < id, t, 1, S, P^N > ∈ P
11         P̃[i] = S[i] OR P̃[i]
12 function updateItemProfile(item vector P^N)
13     for all i ∈ P^N
14         Sum[i] = Integer(P̃[i]) + Integer(P^N[i])
15     for all i ∈ the s highest values in Sum
16         P^N[i] = 1
17 function forward(< id^R, t^R >, item vector S^N, item profile P^N)
18     for all n ∈ Neighbors
19         send < id^R.t^R > with associated S^N and P^N to n
```

As shown in Figure 1 and on lines 14 of Algorithm 1 and 24 of Algorithm 2, a node uses the compact profile both to update the item profile of an item it likes and to compute its obfuscated profile when exchanging clustering information with other nodes. In each of these two cases, the node computes a fresh compact profile as the bitwise OR of the item vectors in its private profile (line 11 of Algorithm 1). This on demand computation allows the compact profile to take into account only the items associated with the current *time window*. It is in fact impossible to remove an item from an existing compact profile. The reason is that compact profile provides a first basic form of obfuscation of the interests of a user through bits collisions : a bit with value 1 in the compact profile of a node may in fact results from any of the liked items whose vectors have the corresponding bit set. By changing the b and d parameters of item vectors, system designers can vary the collision rate between item vectors according to the number of items in the system and thus tune the efficacy of this obfuscation step.

The use of Random Indexing has two clear advantages. First, the presence of bits collisions makes it harder for attackers to identify the items in a given profile. Second, the fixed and small size of bit vectors limits the size of the messages exchanged by the nodes in the system. As evaluated, in Section 7.7, this drastically reduces the bandwidth cost of our protocol.

Item Profile A node never reveals its compact profile. Instead, it injects part of it in the item profiles of the items it likes. Consequently, the item profile of an item gathers the interests of the users that liked the item along its dissemination path. In other words, it reflects the interests of the portion of the network the item has traversed. A parameter s controls how much information from the compact profile nodes include in the item profile.

Algorithm 2: Building obfuscated profile.

```
20 on demend do
21     Algorithm1.buildCompactProfile()
22     buildFilterProfile()
23     for all i ∈ the s highest values in F
24         P*[i] = P̃[i]
25 function buildFilterProfile()
26     for all < id, t, 1, S, P^N >∈ P
27         F[i] = F[i] + Integer(P^N[i])
```

More precisely, let n be a node that liked an item R. When n receives R for the first time, it first computes its compact profile as described above. Then, n builds an integer vector as the bit-by-bit sum of the item profile and its own compact profile (line 14 in Algorithm 1). Each entry in this vector has a value in $\{0, 1, 2\}$: node n chooses the s vector positions with the highest values, breaking ties randomly, and creates a fresh profile for item R by setting the corresponding bits to 1 and the remaining ones to 0. Finally, when n generates the profile for a new item (line 16 in Algorithm 1), it simply sets to 1 the values of s bits from those that are set in its compact profile. This update process ensures that each item profile always contains s bits with value 1.

Filter Profile Nodes compute their filter profiles whenever they need to exchange clustering information with other nodes (line 22 in Algorithm 2). Unlike the other profiles associated with nodes, this profile consists of a vector of integer values and does not represent the interests of a user. Rather it captures the interests of the community of users that have liked similar items. A node computes the value at each position in its filter profile by summing the values of the bits in the corresponding position in the profiles of the items it liked (line 27 in Algorithm 2) in the latest *time window*. This causes the filter profile to record the popularity of each bit within a community of nodes that liked similar items.

Obfuscated Profiles A node computes its obfuscated profile whenever it needs to exchange it with other nodes as part of the clustering protocol. As shown in Figure 1, it achieves this by filtering the contents of its compact profile using its filter profile : this yields a bit vector that captures the most popular bits in the node's community and thus hides its most specific and unique tastes. The fine-grained information contained in the node's private and compact profiles remains instead secret throughout the system's operation.

As shown on line 21 and line 22 of Algorithm 2, a node n computes its obfuscated profile by first generating its compact and filter profiles as described above. Then it selects the s positions that have the highest values in the filter profile, breaking ties randomly, and sets the corresponding bits in the obfuscated profile to the values they have in its compact profile. It then sets all the remaining bits in the obfuscated profile to 0.

The resulting profile has s bits (set at 0 or 1) reflecting the node's compact profile providing a coarse-grained of user interests. Through the value of s, the system designer can control the amount of information that can filter from the compact to the obfuscated profile, and can therefore tune the trade-off between privacy and recommendation quality. It is important to note that the position of the bits at 1 in the obfuscated profile are not correlated to the item vectors of item that user liked but depends on the filter profile. Indeed, only one bit of an item vector

can be selected by the filter profile. This prevents isolated attackers from precisely understanding which news items the node liked as shown in Section 7.5.

5 Randomized Dissemination

As described in Section 3, an attacker can discover the opinions of a user by observing the items she forwards, because of the predictability of the dissemination protocol. We reduce this vulnerability by introducing some randomness in the dissemination process : a node that likes an item will drop it with probability pf, while a node that does not like it will forward it with the same pf. Thanks to this randomization, an attacker cannot deduce with certainty whether a user liked an item from the observation of the items she forwards.

It is important to note that this randomization only affects dissemination. Thus, nodes continue to update their profiles and those of the items as specified in Section 4. In particular, a node will not update the profile of an item it disliked, even if it decides to forward it. Randomization reduces the information leaked during dissemination at the cost of a decrease in accuracy. Indeed, by forwarding disliked items, and dropping liked items, nodes reduce the system's ability to filter and promote relevant content. We evaluate this trade-off in Section 7.5. In the following, we show that this randomization component is differentially private [11].

A randomized algorithm $\mathcal{A}$ is ϵ-differentially private if it produces approximately the same output when applied to two neighboring datasets (*i.e.* which differ on a single element). In the context of dissemination, the datasets that need to be randomized are vectors of user opinions. Given two neighboring vectors of opinions (*i.e.* differing on a single opinion) $o1 \in \mathcal{D}^n$ and $o2 \in \mathcal{D}^n$, we define differential privacy as follows.

Differential privacy [12] A randomized function $\mathcal{F} : \mathcal{D}^n \rightarrow \mathcal{D}^n$ is ϵ-differentially private, if for any pair of neighboring opinion vectors $\mathbf{o1}, \mathbf{o2} \in \mathcal{D}^n$ and for all $\mathbf{t} \in \mathcal{D}^n$:

$$\Pr[\mathcal{F}(\mathbf{o1}) = \mathbf{t}] \leq e^{\epsilon} \cdot \Pr[\mathcal{F}(\mathbf{o2}) = \mathbf{t}]$$

This probability is taken over the randomness of $\mathcal{F}$, while e is the base of the natural logarithm.

In the case of our algorithm, we toss a coin each time the user expresses her opinion about an item in order to decide whether the item should be forwarded. This scheme is known as randomized response [27] : instead of randomizing the output of a function f, we randomize each of its inputs independently. Because these inputs as well as the output values are binary $\in \{0, 1\}$, we can rewrite the above equation as follows.

$$\Pr[f(o) = b] \leq e^{\epsilon} \cdot \Pr[f(1 - o) = b]$$

Our randomization function f flips the opinion o and produces the output $1 - o$ with probability pf. In order to achieve ϵ-differential privacy the value of pf must be such that :

$$1/(e^{\epsilon} + 1) \leq pf \leq 1/2$$

For space reasons, we omit the details of the reasoning leading to this result, as well as the proof of the equivalence between randomized response and Definition 5. Nonetheless they are similar to those in [5].

This algorithm reduces the amount of information an observer gets when receiving an item from a user. Instead of knowing with certainty that the user liked the item, the observer knows that the user liked it with probability $1 - pf$. However, this does not make our solution differentially private. The dissemination component is, but it only ensures ϵ-differential privacy when a user expresses her opinion about an item, not when she generates a new one.

6 Experimental setup

We implemented and extensively evaluated our approach using a real dataset from a user survey. We also compare our solution with a baseline solution with no privacy mechanism, where profiles are exchanged in clear, and a solution that applies a differentially private mechanism both when generating the profiles that users exchange and upon dissemination. We refer to our solution as OPRD (Obfuscation Profile and Randomized Dissemination) in the following.

6.1 Dataset

To evaluate our approach against a real dataset, we conducted a survey on 200 news items involving 120 colleagues and relatives. We selected news items randomly from a set of RSS feeds illustrating various topics (culture, politics, people, sports,...). We exposed this list to our test users and gathered their opinions (like/dislike) on each news item. This provided us with a small but *real* dataset of users exposed to exactly the same news items. To scale out our system, we generated 4 instances of each user and news item in the experiments. While this may introduce a bias, this affects accuracy of both our mechanisms and the solutions we compare against.

6.2 Alternatives

We compare our approach with the two following alternatives.

Cleartext profile (CT) This baseline approach implements the decentralized CF solution presented in Section 2 where user profiles are exchanged in clear during the clustering process. This solution does not provide any privacy mechanism.

Differentially private approach (2-DP) This alternative, denoted by 2-DP in the following, applies randomization both when generating user profiles and during dissemination. Every time a user expresses an opinion about an item, the algorithm inverses it with probability pd : this results in a differentially private clustering protocol and a differentially private dissemination protocol. The latter is similar to our randomized dissemination. However, unlike our solution, 2-DP also applies randomness when generating user profiles. When a user dislikes an item, 2-DP considers this item as liked with a probability pd, thus integrating it in the profile of the user and disseminating it to her neighbors. Conversely, when a user likes an item, 2-DP considers it as disliked with probability pd. In this case, it silently drops it without including it in the user's profile.

2-DP builds user profiles that are structurally similar to our compact profiles. However, they gather the item vectors of the items identified as liked after the randomization of user opinions. This extends the privacy guarantee associated with our dissemination protocol to the profiles of users. This represents a contribution in its own right. For space reasons, we do not include the associated proof. However, it follows a similar intuition than the one presented in Section 5.

As user profiles change over time and are impacted by the dissemination of items, applying a randomization function on cleartext profiles as in [5] is not enough. Iteratively probing the profiles of a user and analyzing the dissemination process could be enough to weaken the privacy guarantee. Instead, 2-DP does not randomize profiles, but it randomizes the opinion of a user on the items she is exposed to. Moreover, it does so independently of the user's opinion on other items.

2-DP uses the output of its randomization function to build user profiles and drive the dissemination. In particular, users use the resulting randomized profiles to compute their clustering

views. We show in Section 7.4 that this introduces a weakness in the context of the decentralized CF scheme considered in this paper.

6.3 Evaluation metrics

6.3.1 Accuracy

We evaluate accuracy along the traditional metrics used in information-retrieval systems : *recall* and *precision*. Both measures are in $[0, 1]$. A recall of 1 means that all interested users have received the item. Yet, a trivial way to ensure a recall of 1 is to send all news items to all users, potentially generating spam. Precision precisely captures the level of spam : a precision of 1 means that all news items reach only users that are interested in them. The F1-Score captures the trade-off between these two metrics and is defined as the harmonic mean of precision and recall [23].

$$Precision = \frac{|\{interested\ \ users\} \cap \{reached\ \ users\}|}{|\{reached\ \ users\}|}$$

$$Recall = \frac{|\{interested\ \ users\} \cap \{reached\ \ users\}|}{|\{interested\ \ users\}|}$$

$$F1 - Score = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

6.3.2 System

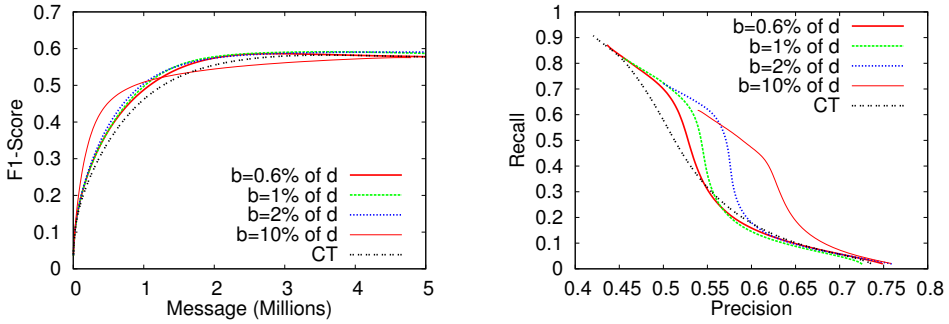
We evaluate the overhead of the system in terms of the network traffic it generates. For simulations, we compute the total number of sent messages. For our implementation, we instead measure the average consumed bandwidth. A key parameter that determines network traffic is the *fanout* of the dissemination protocol, *i.e.* the number of neighbors from the interest-based overlay to which nodes forward each item.

6.3.3 Privacy

We define privacy as the ability of a system to hide the profile of a user from other users. We measure it by means of two metrics. The first evaluates to what extent the obfuscated profile is close to the real one by measuring the similarity between the two. We consider the Jaccard index [23] to measure the similarity between a compact profile and the corresponding obfuscated one. The second measures the fraction of items present in a compact profile out of those that can be predicted by analyzing the presence of item vectors in the corresponding obfuscated profile. As item vectors are public, a malicious user can leverage them to guess the contents of the obfuscated profiles of other users, thereby inferring their interests.

7 Performance evaluation

In this section, we evaluate the ability of our solution to achieve efficient information dissemination while protecting the profiles of its users. First, we show that compacting user profiles, filtering sensitive information, and randomizing dissemination do not significantly affect the accuracy of dissemination when compared to CT, yielding slightly better results than 2-DP. Then we



(a) F1-Score vs number of messages for various b-to-d ratios (b) Precision-recall curve for various b-to-d ratios

FIGURE 2: *Impact of compacting the profiles*

analyze the trade-off between accuracy and privacy and show the clear advantage of our solution in protecting user profiles in the context of a censorship attack. Finally, we show the benefits of our solution in term of network cost. We conducted an extensive evaluation through simulations, and through a real implementation deployed on PlanelLab. In both cases, we randomly select the source of each item among all users.

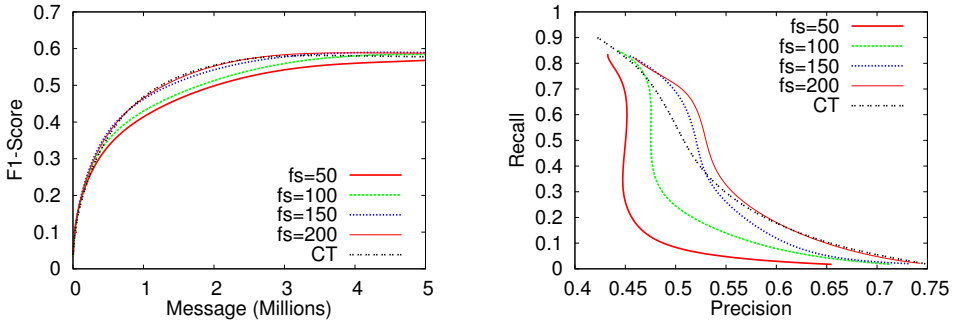
7.1 Compacting profiles

As explained in Section 4.2, our solution associates each item with a (sparse) item vector containing b ones out of d possible positions. When a user likes an item, we add the corresponding item vector to her compact profile by performing a bitwise OR with the current profile. The ratio between b and d affects the probability of having the same bits at 1 in the vectors of two different items in a given compact profile. Figure 2 evaluates its effect on performance.

Figure 2a shows the values of the F1-Score with increasing network traffic for various values of the b-to-d ratio. The points in each curve correspond to a range of fanout values, the fanout being the number of neighbors to which a user forwards an item she likes : the larger the fanout the higher the load on the network. Figure 2b shows instead the corresponding precision-recall curve. Again, each curve reflects a range of fanout values : the larger the fanout, the higher the recall, and the lower the precision.

Interestingly, the larger the values of the b-to-d ratio, the bigger the difference between our solution and CT. When the b-to-d ratio is low, it is very unlikely for any two item vectors to contain common bits at 1. As a result, the performance of our solution closely mimics that of CT. When the b-to-d ratio increases, the number of collisions between item vectors – cases in which two distinct item vectors have common bits at 1 – also increases. This has two interesting effects on performance.

The first is that the F1-Score increases faster with the fanout and thus with the number of messages : the $b = 10\%$ curve climbs to an F1-Score of 0.4 with less than $400k$ messages. The curve on Figure 2b shows that this results from a higher recall for corresponding precision values (bump in the $b = 10\%$ curve). The high probability of collisions between item vectors results in some profiles being similar even if they do not contain many common items. The resulting more clustered topology facilitates dissemination and exposes nodes to more items. This makes it easier for them to discover real similar interests and therefore good neighbors even at low fanout



(a) F1-Score vs number of messages for various filter sizes (fs)

(b) Precision-recall curve for various size of filters (fs)

FIGURE 3: *Impact of filtering sensitive information*

values.

However, the second effect is that the maximum F1-Score attained by the protocol with a large b-to-d ratio (to the right of Figure 2a) stagnates to lower values. Figure 2b clarifies that this results from lower recall values, as indicated by the left endpoints of the curves corresponding to high values of b. The reason for this behavior is that the clustered topology resulting from large values of b, advantageous with small fanout values (small number of messages), ultimately inhibits the dissemination of items to large populations of users. This effect is even more prominent when considering values of b for which compact profiles contain a vast majority of bits set to 1 (not shown in the plot).

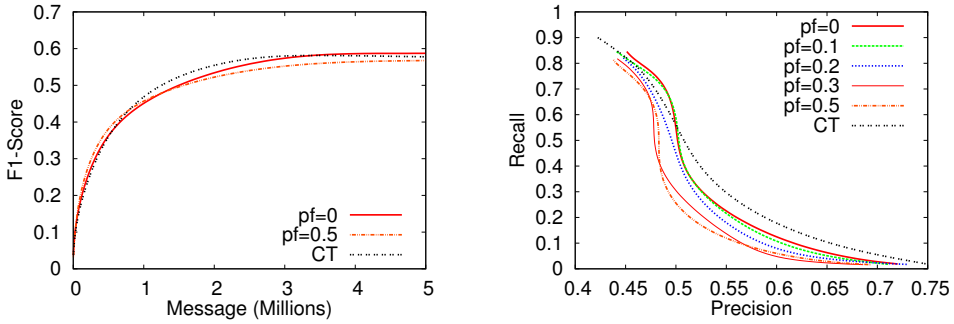
In the following, we set d to 500 and b to 5 for our evaluations. System designers should set these parameters according to the rate at which items are generated to achieve the desired rate of collisions in compact profiles.

7.2 Filtering sensitive information

In our solution, the size of the filter defines how much information from the compact profile appears in the obfuscated profile. The larger the filter, the more the revealed information. Figure 3a depicts the F1-Score as a function of the number of messages. Clearly, performance increases with the size of the filter. The precision-recall curve in Figure 3b shows that this results mainly from an increase in precision. The important aspect is that both plots highlight that a filter of 200 bits (*e.g.* 40% of the compact profile) achieves performance values similar to those of a system using full profiles.

7.3 Randomizing the dissemination

We now evaluate the impact of randomizing the dissemination process in addition to the obfuscation mechanism evaluated above (previous results were obtained without randomization). This removes the predictive nature of dissemination. Figure 4a shows the F1-Score for our solution using a filter size of 150 and several values for pf. Performance decreases slightly as we increase the amount of randomness (for clarity, we only show $pf = 0$ and $pf = 0.5$, the other curves being in between). Figure 4b shows that increasing pf results mostly in a decrease in precision.



(a) F1-Score vs number of messages for various pf ($fs = 150$)

(b) Precision-recall curve for various pf ($fs = 150$)

FIGURE 4: *Impact of obfuscating profiles and randomizing dissemination*

7.4 Evaluating 2-DP

In this section, we evaluate the 2-DP alternative defined in Section 6.2. 2-DP reverses the behavior of users with a probability, pd, that affects both the construction of user profiles and the dissemination process. This differs from our solution, which only uses a differentially private dissemination protocol.

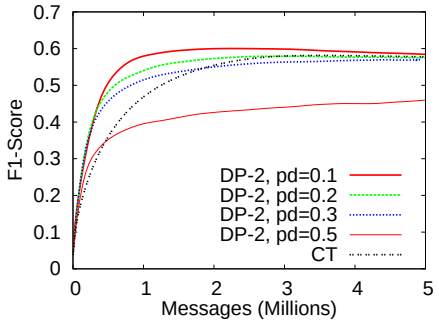
Figure 5a shows the F1-Score of 2-DP versus network traffic for various values of pd. While for $dp = 0$, 2-DP is equivalent to CT, performance at low fanout values increases for $dp = 0.1$, but decreases again for larger values of dp. A small amount of randomness proves beneficial and allows the protocol to achieve a better compromise between recall and precision at low fanouts. This effect, however, disappears when the number of messages increases at high fanouts. Too much randomness, on the other hand, causes a drastic decrease in the F1-Score. Figure 5b shows that randomness induces an increase in recall with respect to CT and a decrease in precision. The former dominates with low values of pd while the latter dominates for high values.

Figure 5c compares the F1-Score of OPRD using a filter of size of 150 and a pf value of 0.3, with that of CT and 2-DP using a pd of 0.3. We observe that above $2M$ messages, our solution provides slightly better F1-Score values than 2-DP. Overall, however, the best performances of the two approaches are comparable. In the following, we show that this is not the case for their ability to protect user profiles.

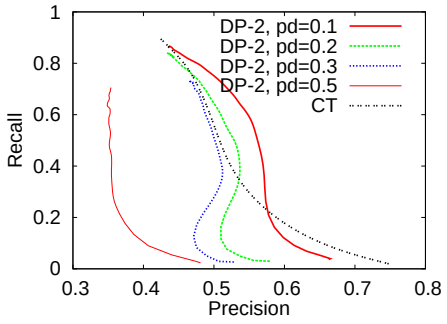
7.5 Privacy versus accuracy

We evaluate the trade-off between the privacy, measured as the ability to conceal the exact profiles of users, and the accuracy of both our solution and 2-DP. In our solution, this trade-off is controlled by two parameters : the size of the filter, and the probability pf to disseminate a disliked item. 2-DP controls this trade-off by tuning the probability pd to switch the opinion of the user, impacting both profile generation and the dissemination process.

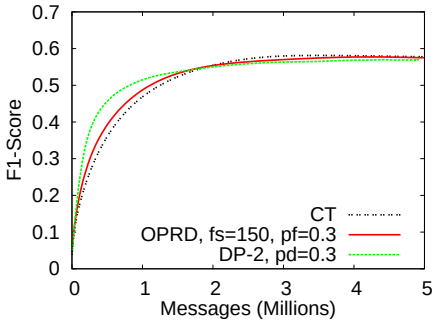
Figure 6a compares the recommendation performance by measuring the F1-Score values for various filter sizes. The x-axis represents the evolution of the probabilities pf, for our solution, and pd, for 2-DP. We show that the F1-Score of 2-DP decreases faster than ours. The F1-Score of 2-DP with a pd of at least 0.2 is smaller than that of our solution with a filter size greater than 100. In addition, revealing the least sensitive 10% of the compact profile ($fs = 50$) yields better performance than 2-DP with $pd \geq 0.3$.



(a) 2-DP : F1-Score vs number of messages for various *pd*



(b) 2-DP : Precision-recall curve for various *pd*



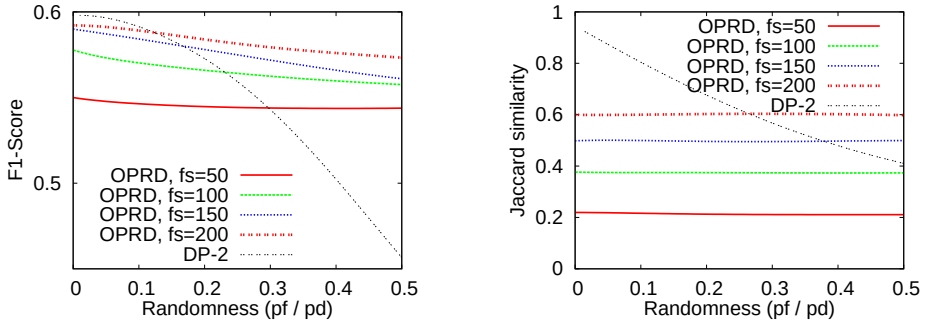
(c) OPRD vs 2-DP : F1-Score vs number of messages

FIGURE 5: *Differential private alternative*

Figure 6b measures the level of privacy as the similarity between the compact profile and the obfuscated profile using the Jaccard index : lower similarity implies more privacy. Results show the flexibility of our solution. As our randomized dissemination protocol hardly impacts the obfuscated profile, our results are almost independent of pf. 2-DP sees instead its similarity decrease with increasing pd. With $pd = 0.3$, 2-DP yields a similarity value of about 0.55 with an F1-Score (from Figure 6a) of 0.56. Our approach, on the other hand yields the same similarity value with a filter size between $150 < fs < 200$, which corresponds to an F1-Score value of about 0.58.

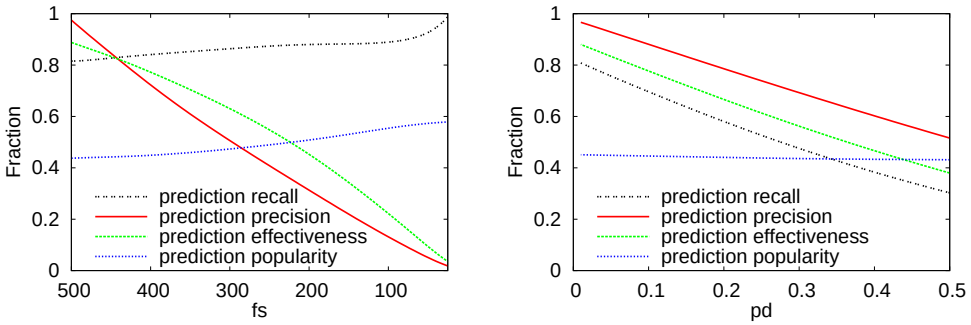
Figure 7, instead, assesses privacy by measuring if the items in a user's real profile can be predicted by analyzing the item vectors in profile exchanged with neighbors. Note that in the 2-DP, the real profile is the one that would exist without random perturbations. We evaluate this aspect by measuring the recall and the precision of each prediction. Prediction recall measures the fraction of correctly predicted items out of those in the compact profile. Prediction precision measures the fraction of correct predictions out of all the prediction attempts. For our solution, in Figure 7a, we use a $pf = 0.2$ to control the randomized dissemination, and vary the filter size. For 2-DP (Figure 7b), we instead vary pd.

The plots show that while predictions on our approach can be fairly precise, they cover only a very small fraction of the compact profile with reasonable values of fs. With $fs = 200$, which gives similar performance as using the entire compact profile (Figure 3), the prediction recall is



(a) Accuracy : OPRD vs 2-DP (b) Similarity between public and real profiles

FIGURE 6: *Randomness vs performance and level of privacy*



(a) Prediction with OPRD, $pf = 0.2$ (b) Prediction with 2-DP

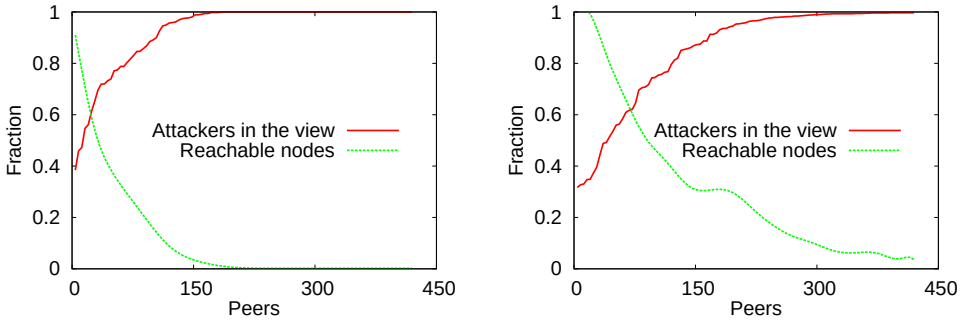
FIGURE 7: *Profile prediction*

of about one third. In contrast, 2-DP exposes a higher number of items from the compact profile. With $pd = 0.2$ the prediction recall is of 0.8 with a prediction precision of 0.6. The curves for prediction effectiveness, computed as the harmonic mean of recall and precision, further highlight our approach's ability to strike an advantageous balance between privacy and recommendation performance.

The two plots also show the average popularity of the predicted items. It is interesting to observe that with small enough filters, we predict the most popular items, which are arguably the least private. Finally, we also observe that the compact profile itself provides a small protection to the prediction of items due to its inherent collision rate. With a filter of size 500 (*e.g.* with no difference between the compact and the public profile), the error rate is equal to 0.15.

7.6 Illustration : Resilience to censorship attack

Exposing user profiles enables potential censorship attacks. In this section, we evaluate the resilience of our obfuscation mechanism against censorship by implementing a simple eclipse attack [21]. In this attack, a coalition of censors mirrors the (obfuscated) profile of a target node in order to populate its clustering view. This is turn isolates it from the remaining nodes since its only neighbors are all censors. If the user profiles are exposed in clear, the profile of the censors



(a) Censorship : 2-DP, $pd = 0.2$ (b) Censorship : OPRD, $fs = 125$, $pf = 0.2$

FIGURE 8: *Resilience to censorship*

matches exactly that of the target node : this gives censors a very high probability to enter its view. Once the censors have fully populated the target node's view, they simply intercept all the messages sent by the target node, preventing their dissemination. To evaluate the efficiency of this attack, we use two different metrics. The first is the poisoning rate of the target's clustering view by attackers. The second one is the fraction of honest nodes (*e.g.* not censors) in the system that the target can reach when it sends an item.

We ran this attack for each user in the dataset. The x-axis represents the users in the experiment sorted by their sensitivity to the attack. Figure 8a and Figure 8b depict the results obtained with a cluster size of 50 and 50 censors (we observe similar results independently of the cluster size). In addition, this experiment uses a filter of 125 and $pf = 0.2$ for our solution, and $pd = 0.2$ for 2-DP. We can clearly see that 2-DP is not effective in preventing censorship attacks : only 150 nodes have a poisoning rate lower than 1. This is due to the fact that in order to ensure differential privacy, 2-DP, randomizes the profiles once so that they cannot be iteratively probed. However, this forces 2-DP to use the same profile internally and during the exchanges for similarity computation. Therefore 2-DP exhibits the exact same vulnerability as CT. The profiles of the censors can trivially match the target node.

Our approach is much more resilient to this specific censorship attack. For instance, it is almost impossible for censors to intercept all the messages sent by the target and only a third of the nodes have a fully poisoned clustering view. The obfuscated profile only reveals the least sensitive information to other nodes : censors only mirror a coarse-grained sub part of the target's profile. As a consequence, they are more likely to have a profile similar to users with similar interests than to match the one of the target. This observation is confirmed by Figure 6b which shows the difference in terms of similarity between the obfuscated profile and the compact profile of users. The resilience of OPRD to this censorship attack is driven by the size of the obfuscation filter, the smaller the filter, the more resilient the protocol.

7.7 Bandwidth consumption

We also conducted experiments using our prototype with 215 users running on approximately 110 PlanetLab nodes in order to evaluate the reduction on the network cost due to the dimension reduction of our profiles. The results in terms of F1-Score, recall, and precision closely mimic those obtained with our simulations and are therefore omitted. Table 1, on the other hand, shows the cost of our protocols in terms of bandwidth : our obfuscation mechanism is effective in

Fanout	2	4	6	8	10	15	20
CT	1.8	3.0	4.4	6.5	8.2	12	14
OPRD	0.8	1.1	1.5	1.7	2.7	2.8	4.1

TABLE 1: *Bandwidth usage in kbps per node in PlanetLab*

reducing the bandwidth consumption of decentralized collaborative filtering. The cost associated with our obfuscated solution is about one third of that of the solution based on cleartext profiles.

7.8 Summary

While 2-DP ensures that it is impossible to extract more information than the one revealed in the user profile, our solution offers a more advantageous trade-off between accuracy and privacy. For instance, OPRD achieves an F1-Score = 0.58 with $fs = 150$ and $pf = 0.2$, while 2-DP achieves the same with $pd = 0.2$. With these parameters, the profile of 2-DP exposes more of the user's interest (*i.e.* similarity with real profile of 0.7 for 2-DP against 0.5 for OPRD) and results in higher prediction effectiveness. (*i.e.* 80% against 26% for OPRD). In addition, as the perturbation injected in the user profiles depends on similar users in term of interests, OPRD is more resilient to censorship attacks. However, OPRD does not provide as strong guarantees as 2-DP, especially against advanced active attackers or massive groups of colluders that could leverage information collected in the system to estimate the interests of users.

8 Related work

Privacy is important in many applications. Several approaches [3, 19, 20] use randomized masking distortion techniques to preserve the privacy of sensitive data. However, [14] shows that the predictable structure in the spectral domain of the random distortion can seriously compromise privacy. In the same vein, [16] shows that the variances of the random noises have an important impact on the possibility to filter noise from the original data. In our solution, instead of adding perturbation to user profiles, we exchange with other users a coarse-grain version of this profile only revealing its least sensitive information. The perturbation applied on the item profile is not random and depends on the interest of users. As a consequence, separating privacy sensitive information from the distortion becomes more difficult.

[2] designed a statistical measure of privacy based on differential entropy. While this measure can be used to evaluate privacy in a collaborative system [20], it is however difficult to identify the meaning of its value and the implication on the sensitive data. Differential privacy was considered in [11, 13]. In a distributed settings, [5] proposed a differentially private protocol to measure the similarity between peers. While this solution works well in case of static profiles, its differential privacy is not preserved when profiles are dynamic as in recommendation systems. In addition, still in the context of recommendation systems, [18] highlights the trade-off between privacy and accuracy.

Other approaches [9, 8] exploit homomorphic encryp-tion based approaches in a P2P environment to secure multi- party computation techniques. Similarly, [4] proposes an architecture for privacy preserving CF by replacing the single server providing the service by a coalition of trusted servers.

9 Conclusions

We proposed a simple and efficient mechanism to control the trade-off between privacy and recommendation quality in decentralized collaborative filtering. Our mechanism relies on two components : (*i*) an original obfuscation mechanism revealing only the least sensitive information in the profiles of users, and (*ii*) a randomization-based dissemination algorithm ensuring differential privacy during the dissemination process. In addition, we also proposed a fully differentially private alternative which can be viewed as a contribution in its own right. Interestingly, our mechanism achieves similar results as this differentially private alternative in terms of recommendation quality while protecting the sensitive information of users more efficiently and being more resilient to censorship attacks.

Références

[1] Reference hidden for blind review as an entry in the bibliography.

[2] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *PODS*, 2001.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD*, 2000.

[4] W. Ahmad and A. Khokhar. An architecture for privacy preserving collaborative filtering on web portals. In *IAS*, 2007.

[5] M. Alaggan, S. Gambs, and A-M. Kermarrec. BLIP : Non-interactive Differentially-Private Similarity Computation on Bloom Filters. In *SSS*, 2012.

[6] M. Bertier, D. Frey, R. Guerraoui, A.M. Kermarrec, and V. Leroy. The gossple anonymous social network. In *Middleware*, 2010.

[7] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 1970.

[8] J. Canny. Collaborative filtering with privacy. In *SP*, 2002.

[9] J. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR*, 2002.

[10] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization : scalable online collaborative filtering. In *WWW*, 2007.

[11] C. Dwork. Differential privacy : a survey of results. In *TAMC*, 2008.

[12] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*. 2006.

[13] A. Haeberlen, B. C. Pierce, and A. Narayan. Differential privacy under fire. In *SEC*, 2011.

[14] Z. Huang, W. Du, and B. Chen. Deriving private information from randomized data. In *SIGMOD*, 2005.

[15] P. Kanerva, J. Kristoferson, and A. Holst. Random indexing of text samples for latent semantic analysis. In *CCSS*, 2000.

[16] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *ICDM*, 2003.

[17] G. Linden, B. Smith, and J. York. Amazon.com recommendations : Item-to-item collaborative filtering. *IEEE Internet Computing*, 2003.

[18] A. Machanavajjhala, A. Korolova, and A. D. Sarma. Personalized social recommendations : accurate or private. *VLDB*, 2011.

[19] H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *ICDM*, 2003.

[20] H. Polat and W. Du. Svd-based collaborative filtering with privacy. In *SAC*, 2005.

[21] A. Singh, M. Castro, P. Druschel, and A. Rowstron. Defending against eclipse attacks on overlay networks. In *SIGOPS*, 2004.

[22] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

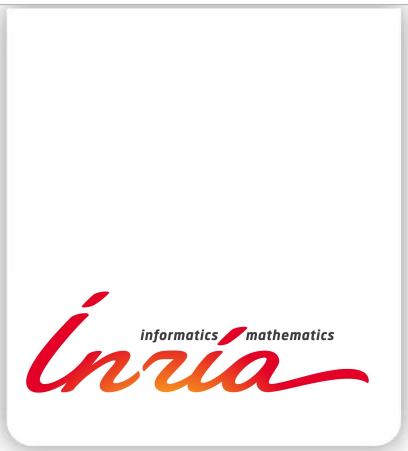
[23] C. J. van Rijsbergen. *Information retrieval*. Butterworth, 1979.

[24] S. Voulgaris, D. Gavidia, and M. v. Steen. Cyclon : inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 2005.

[25] S. Voulgaris and M. v. Steen. Epidemic-style management of semantic overlays for content-based searching. In *Euro-Par*, 2005.

[26] M. Wan, A. Jönsson, C. Wang, L. Li, and Y. Yang. A random indexing approach for web user clustering and web prefetching. In *PAKDD*, 2012.

[27] Stanley L. Warner. Randomized response : a survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309) :63–69, March, 1965.



RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399