Privacy-Preserving Distributed Collaborative Filtering

Antoine Boutet[1(✉)], Davide Frey[1], Rachid Guerraoui[2], Arnaud Jégou[1], and Anne-Marie Kermarrec[1]

[1] INRIA Rennes, Rennes, France
{antoine.boutet,davide.frey,arnaud.jegou,Anne-marie.kermarrec}@inria.fr
[2] EPFL, Lausanne, Switzerland
rachid.guerraoui@epfl.ch

Abstract. We propose a new mechanism to preserve privacy while leveraging user profiles in distributed recommender systems. Our mechanism relies on (i) an original obfuscation scheme to hide the exact profiles of users without significantly decreasing their utility, as well as on (ii) a randomized dissemination protocol ensuring differential privacy during the dissemination process.

We compare our mechanism with a non-private as well as with a fully private alternative. We consider a real dataset from a user survey and report on simulations as well as planetlab experiments. We dissect our results in terms of accuracy and privacy trade-offs, bandwidth consumption, as well as resilience to a censorship attack. In short, our extensive evaluation shows that our twofold mechanism provides a good trade-off between privacy and accuracy, with little overhead and high resilience.

1 Introduction

Collaborative Filtering (CF) leverages interest similarities between users to recommend relevant content [19]. This helps users manage the ever-growing volume of data they are exposed to on the Web [8]. But it also introduces a trade-off between ensuring user privacy and enabling accurate recommendations. Decentralized collaborative filtering partially addresses this trade-off by removing the monopoly of a central entity that could commercially exploit user profiles. However, it introduces new privacy breaches: users may directly access the profiles of other users. Preventing these breaches is the challenge we address in this paper. We do so in the context of a news-oriented 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 components of a decentralized user-based CF system: clustering and dissemination. Clustering consists in building an interest-based topology, implicitly connecting users with similar preferences: it computes the similarity between profiles, capturing the opinions of users on the items they have been exposed to. The dissemination protocol propagates the items along the resulting topology.

G. Noubir and M. Raynal (Eds.): NETYS 2014, LNCS 8593, pp. 169–184, 2014.
DOI: 10.1007/978-3-319-09581-3_12

Our obfuscation scheme prevents user machines from exchanging their exact profiles while constructing the interest-based topology. We compute similarities using coarse-grained obfuscated versions of user profiles that reveal only the least sensitive information. To achieve this, we associate each disseminated item with an *item profile*. This profile aggregates information from the profiles of users that liked 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. We use this information to construct filters that identify the least sensitive parts of user profiles: those that are the most popular among users with similar interests. Albeit lightweight, our *obfuscation* scheme prevents any user from knowing, with certainty, the exact profile of another user. Interestingly, we achieve this without significantly hampering the quality of recommendation: the obfuscated profile reveals enough information to connect users with similar interests.

Our dissemination protocol ensures differential privacy [9]. 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 obtain differential privacy by introducing randomness in the dissemination of items. This prevents malicious players from guessing the interests of a user from the items she forwards.

We compare our mechanism with a non-private baseline as well as with an alternative solution that applies differential privacy to the entire recommendation process. We consider a real dataset from a user survey and report on simulations as well as planetlab experiments. We dissect our results in terms of accuracy and privacy trade-offs, bandwith consumption, as well as resilience to a censorship attack. Our extensive evaluation shows 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.

2 Setting

We consider a decentralized news-item recommender employing user-based collaborative filtering (CF). Its architecture relies on two components: *user clustering* and *item dissemination*. We aim to protect users from *privacy threats*.

User clustering aims at identifying the k nearest neighbors of each user[1]. It maintains a dynamic interest-based topology consisting of a directed graph

[1] We use the terms 'node' and 'user' interchangeably to refer to the pair 'user/machine'.

$G(U, E)$, 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. 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. We focus on systems based on binary ratings: a user either *likes* or *dislikes* an item. The interest-based topology exploits two gossip protocols running on each node. The lower-layer random-peer-sampling (RPS) [22] protocol ensures connectivity by maintaining a continuously changing random graph. The upper-layer clustering protocol [6,23] starts from this random graph and quickly provides each node with its k closest neighbors according to a similarity metric. Several similarity metrics have been proposed [21], we use the Jaccard index in this paper.

Item dissemination exploits the above clustering scheme to drive the dissemination. When a user generates a new item or 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 drops it.

Privacy Threats. 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 the decentralized item recommender considered, malicious nodes can extract information in two ways: (i) from the profiles they exchange with other nodes (profiles contain information about the interests of users); and (ii) from the predictive nature of the dissemination (a node sends an item only when it likes it). We consider the Honest-But-Curious adversary model [11] where malicious nodes can collude to predict interests from received profiles but cannot cheat in the protocol. In Sect. 6.6, we also consider attackers modifying their obfuscated profiles to control their location in the interest-based topology (*i.e.* their clustering views).

3 Obfuscation Protocol

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. For clarity, this Section describes a simplified version of our obfuscation protocol. Section 4 completes this description with features required by our differentially-private dissemination scheme. Figure 1 gives an overview of the complete protocol. For space reason, we omit the pseudocode of the algorithms (available in [5]).

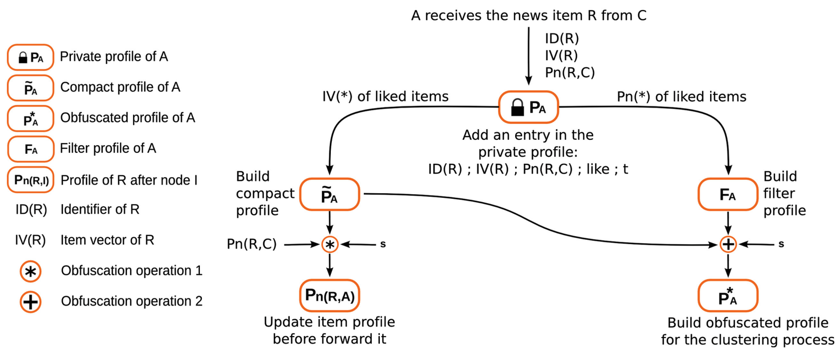


Fig. 1. Simplified information flow through the protocol's data structures.

3.1 Overview

Our protocol relies on random indexing, an incremental dimension reduction technique [14,24]. To apply it in our context, we associate each item with an *item vector*, a random signature generated by its source node. An *item vector* consists of a sparse d-dimensional bit array. To generate it, the source of an item randomly chooses $b << d$ distinct array positions and sets the corresponding bits to 1. It then attaches the *item vector* to the item before disseminating it.

Nodes use item vectors when recording information about items in their obfuscated profiles. Let us consider a node A that receives an item R from another node C. Figure 1 depicts the data flow through the protocol's data structures. When receiving R, node A records whether it likes or dislikes the item in its *private profile*. A node never shares its private profile. It only uses it as a basis to build an *obfuscated profile* whenever it must share interest information with other nodes in the clustering process. Nodes remove the items whose timestamps are outside the latest *time window*. This ensures that all profiles reflect the current interests of the corresponding nodes.

Let us now assume that A receives an item R and likes it. After updating its private profile, A updates the item profile of R before forwarding it to other nodes. This corresponds to the operations on the left branch of Fig. 1. A extracts the items it has liked from its *private profile* and combines the corresponding item vectors into a data structure called *compact profile*. This introduces some uncertainty because different sets of liked items may result in the same *compact profile*. Then A updates the *item profile* of R. This consists of a bitmap that aggregates the compact profiles of the nodes that liked R. To update it, A combines its own *compact profile* and R's old *item profile*. This makes R's *item profile* an obfuscated summary of the interests of the nodes that like R.

The right branch of Fig. 1 shows how a node, A, builds its *obfuscated profile* when required by the clustering process. First, A creates a *filter profile* that aggregates the information contained in the *item profiles* of the items it liked. Then, it uses this filter to identify the bits from its *compact profile* that will appear in its *obfuscated profile*. These consist of the most popular bit positions among the nodes that liked the same items as A did. This has two advantages. First, using the most popular bits makes A's *obfuscated profile* likely to overlap with those of similar nodes. Second, these bits carry less information than less popular ones, which makes them preferable in terms of privacy.

3.2 Profile Updates

Private Profile. A node updates its private profile whenever it generates a new item or receives an item from another node. In either case, the node inserts a new tuple into its private profile. This tuple contains the item identifier, its timestamp (indicating when the item was generated), a score (1 if the node liked the item, 0 otherwise), its item vector, and the item profile upon receipt. Locally generated items count as liked and have an empty "item profile upon receipt".

Compact Profile. Unlike private profiles, which contain item identifiers and their associated scores, the compact profile stores liked items in the form of a d-dimensional bit array. As shown in Fig. 1, 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 of all the liked items in its private profile.

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 the compact profile provides a first basic form of obfuscation of the interests of a user through bit collisions: a bit at 1 in the compact profile of a node may in fact result from any of the liked items whose vectors have the corresponding bit set.

Compact profiles bring two clear benefits. First, the presence of bit 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 Sect. 6.7, this drastically reduces the bandwidth cost of our protocol.

Item Profile. A node never reveals its compact profile. Instead, it injects part of it into the item profiles of the items it likes. Consequently, the item profile of an item aggregates the interests of the users that liked the item along its dissemination path. A parameter s controls how much information from the compact profile nodes include in the item profile.

Let n be a node that liked an item R. When receiving R for the first time, n computes its compact profile as described above. Then, before forwarding R, n builds an integer vector as the bit-by-bit sum of R's item profile and n's own compact profile. 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, it simply sets to 1 the values of s bits from those that are set in its compact profile. This update process ensures that the item profile of each forwarded item always contains s bits with value 1.

Filter Profile. Nodes compute their filter profiles whenever they need to exchange clustering information with other nodes. 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 liked items. This causes the filter profile to record the popularity of each bit within a community of nodes that liked similar items.

Obfuscated Profiles. As shown in Fig. 1, a node builds its obfuscated profile by filtering the contents of its compact profile through the filter profile. This yields a bit vector that captures the most popular bits in the node's community and thus hides the node's most specific and unique tastes. The node selects the s positions that have the highest values in its 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.

4 Randomized Dissemination

An attacker can discover the opinions of a user by observing the items she forwards (Sect. 2). We address this vulnerability through our second contribution: a differentially-private randomized dissemination protocol.

The key idea of our protocol is to randomize the forwarding decision: a node that likes an item drops it with probability pf, while a node that does not like it forwards it with the same pf. This prevents an attacker from acquiring certainties about a user's interests by observing which items she forwards. However, the attacker could still learn something from the content of the associated *item profiles* (nodes modify the item profiles of the items they like). To ensure that the whole dissemination protocol does not expose any non-differentially-private information, we therefore randomize not only forwarding actions, but also the *item profiles* associated with forwarded items. This requires us to modify the protocol described in Sect. 3 as follows.

First, we modify the *private profile*. For each item, not only do we store whether the node liked or disliked it, but we also add a new field: the *randomized decision*. This field stores the forwarding decision taken as a result of the randomization process (1 for forward and 0 for drop).

We then introduce a new *randomized compact profile* (as shown in Fig. 2). The node fills this profile analogously to the compact profile but it uses the *randomized decision* instead of its actual opinion on the item. The node iterates through all the items for which the randomized decision is 1 and integrates their item vectors into the *randomized compact profile* using the same operations described for the non-randomized one.

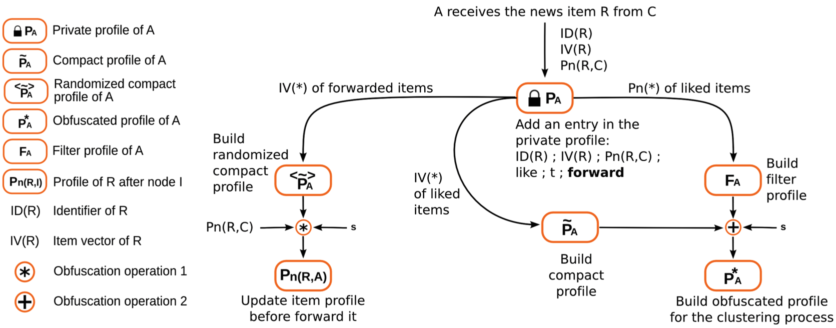


Fig. 2. Complete information flow through the protocol's data structures.

Finally, the node updates the *item profile* of an item when it decides to forward it as a result of randomization, regardless of whether it likes it or not. Moreover, the node performs this update as described in Sect. 3.2 except that the node uses its *randomized compact profile* instead of its *compact profile*. Nodes still use their non-randomized *compact profile* when choosing their neighbors. In this case, they compare their *compact profile* with the *obfuscated profiles* of candidate neighbors.

The above modifications guarantee that the actual content of the *compact profile* never leaks during dissemination. This guarantees that our dissemination protocol is *differentially private* [9]. Roughly speaking, a randomized algorithm $\mathcal{A}$ is ϵ-differentially private if it produces approximately the same output when applied to two neighboring datasets [10] (*i.e.* which differ on a single element). In the context of dissemination, the datasets to randomize are vectors of user opinions. For space reasons, we omit the proofs, which can be found in [5].

This algorithm bounds 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. In the latter case the user always forwards the item.

5 Experimental Setup

Dataset. We conducted a survey on 200 news items involving 120 colleagues and relatives. We selected news items from a set of RSS feeds illustrating various topics. We exposed each of them to our test users and gathered their opinions (like/dislike). 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, it does so for both our mechanism and the two solutions we compare against.

Alternatives. We compare our approach with the following alternatives.

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

Differentially private approach (2-DP): This alternative, denoted by 2-DP in the following, uses the *randomized compact profile* both for clustering and for dissemination. In other words, it applies randomization during the entire recommendation process. In particular, nodes leverage their randomized compact profiles to compute their clustering views. Every time a user expresses an opinion about an item, 2-DP inverses it with probability pd: this results in a differentially private clustering protocol and a differentially private dissemination protocol.

2-DP extends the privacy guarantee provided by our dissemination protocol to the management of interest profiles. Section 6.6 shows that 2-DP remains more vulnerable to censorship attacks than our solution.

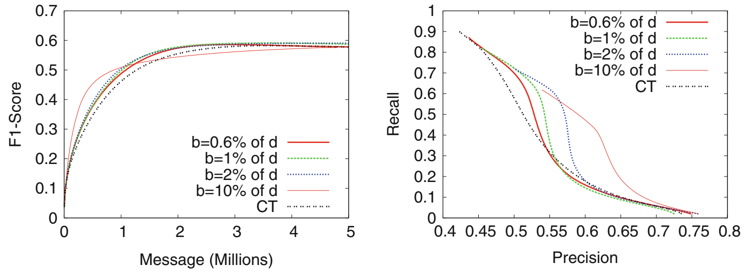
Recommendation Quality. We evaluate recommendation using *recall* and *precision*. Both measures are in $[0, 1]$. A recall of 1 means that all interested users have received the item. Yet, a trivial way to ensure a recall of 1 is to send all news items to all users, potentially generating spam. Precision 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 [21] as their harmonic mean.

Overhead. 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.

Privacy. We measure privacy as the ability of a system to hide the profile of a user from other users. We capture it by means of two metrics. The first evaluates to what extent the obfuscated profile is close to the real one by measuring the overlap rate between the two using the Jaccard index. 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, malicious users can leverage them to guess contents of the obfuscated profiles of other users, thereby inferring their interests.

6 Performance Evaluation

We evaluate the ability of our solution to achieve efficient information dissemination while protecting the profiles of its users. We consider both simulations, and a real implementation deployed on PlanelLab. In both cases, we randomly select the source of each item among all users. We refer to our solution as OPRD (Obfuscation Profile and Randomized Dissemination) in the following.



(a) F1-Score vs messages (b) Precision-recall curve

Fig. 3. Impact of compacting the profiles (various b-to-d ratios)

6.1 Compacting Profiles

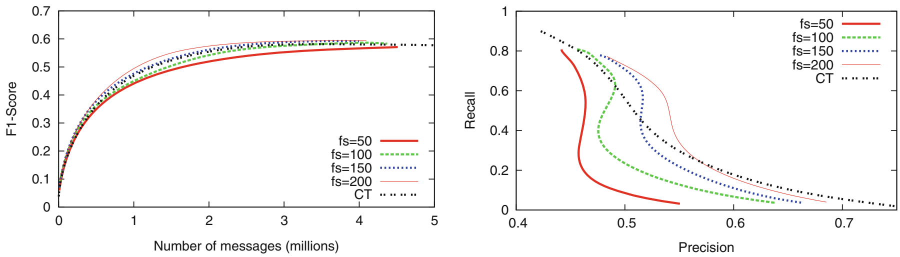
As explained in Sect. 3.2, our solution associates each item with a (sparse) item vector containing b 1's out of d possible positions. When a user likes an item, we add the corresponding item vector to her compact profile by performing a bitwise OR with the current profile. The ratio between b and d affects the probability of having two items sharing bits at 1 in their vectors, which in turn affects the accuracy of the similarity computation between users. Figure 3 evaluates its effect on performance.

Figure 3a shows the values of the F1-Score depending on 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 3b 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 b-to-d ratio, the bigger the difference between our solution and CT. With a low b-to-d ratio, it is 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 Fig. 3b 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 user profiles being similar even though they do not contain many common items. This leads to a topology in which users are less clearly clustered, and in which the items can be disseminated more easily, which explains the high recall value.

The second effect is that the maximum F1-Score attained by the protocol with a large b-to-d ratio (to the right of Fig. 3a) stabilizes at lower values. Figure 3b clarifies that this results from a lower maximum recall, as indicated



(a) F1-Score vs messages (b) Precision-recall curve

Fig. 4. Impact of filtering sensitive information (various filter sizes, fs)

by the left endpoints of the curves corresponding to high values of b. The artificial similarities caused by a large b—advantageous with small fanout values (small number of messages)—also create false clusters that ultimately inhibit the dissemination of items to large populations of users. This effect is even more prominent with values of b that set a vast majority of the bits in compact profiles to 1 (not shown in the plot).

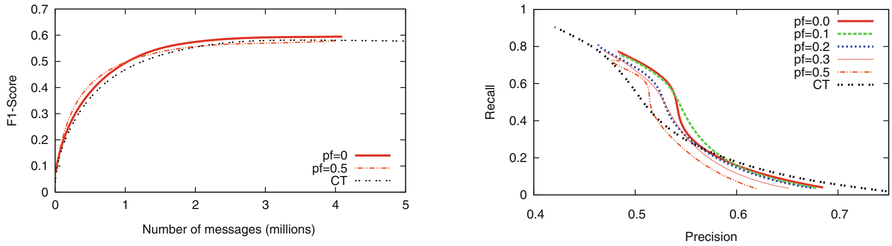
In the following, we set d to 500 and b to 5 for our evaluations. The values assigned to b and d should be computed depending on the expected number of items per user profile. Explanations about the computation of these values are outside of the scope of this paper, but are similar to those that relate the number of hash functions and the size of a bloom filter [20].

6.2 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 4a depicts the F1-Score as a function of the number of messages. The performance increases with the size of the filter. Figure 4b shows that this variation comes from the fact that precision strongly decreases when the filter size decreases.

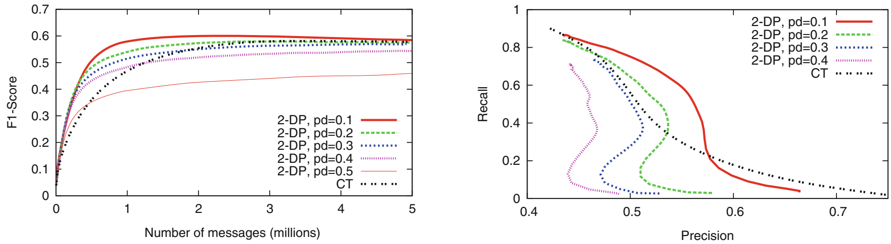
6.3 Randomizing the Dissemination

We now evaluate the impact of randomizing the dissemination process in addition to the obfuscation protocol evaluated above (the previous results were obtained without randomization). Figure 5a shows the F1-Score for our solution using a filter size of 200 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 5b shows that increasing pf results mostly in a decrease in precision.



(a) F1-Score vs messages for various pf (b) Precision-recall curve for various pf

Fig. 5. Impact of obfuscating profiles and randomizing dissemination ($fs = 200$)



(a) F1-Score vs messages for various pd (b) Precision-recall curve for various pd

Fig. 6. Impact of the randomization for 2-DP

6.4 Evaluating 2-DP

2-DP reverses the opinions of users with a probability, pd, that affects both the construction of user profiles and the dissemination process. This differs from our solution in which only the dissemination is randomized.

Figure 6a shows the F1-Score of 2-DP versus network traffic for various values of pd. Performance strongly increases at low fanout values for $dp = 0.1$, but decreases for larger values. A small amount of randomness proves beneficial and allows the protocol to disseminate items more effectively with a low fanout. This effect, however, disappears at high fanouts. Very high values of pd on the other hand cause a drastic decrease in the F1-Score. Figure 6b shows that increasing randomness leads to a strong decrease in precision.

Figure 7 compares the F1-Score of OPRD using a filter of size of 200 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.

6.5 Privacy Versus Accuracy

We evaluate the trade-off between privacy, measured as the ability to conceal the exact profiles of users, and accuracy for both OPRD and 2-DP. OPRD controls

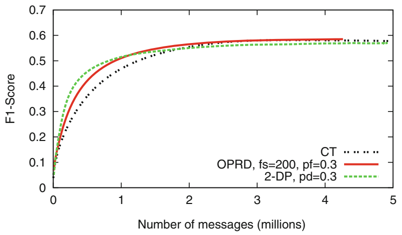
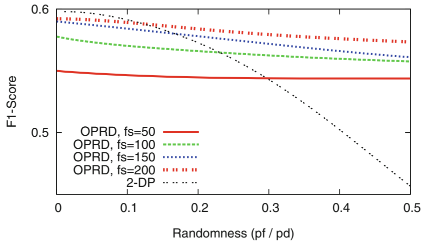
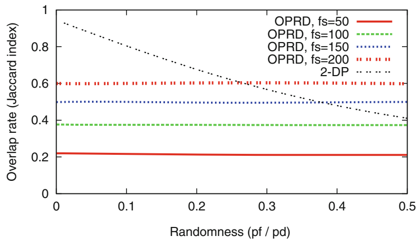


Fig. 7. OPRD vs 2-DP: F1-Score vs number of messages



(a) Accuracy: OPRD vs 2-DP



(b) Public and real profiles overlap

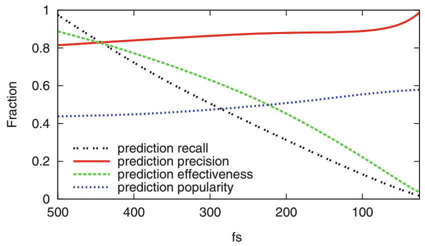
Fig. 8. Randomness vs performance and level of privacy

this trade-off with two parameters: the size of the filter, and the probability pf. 2-DP controls it by tuning the probability pd to switch the opinion of the user.

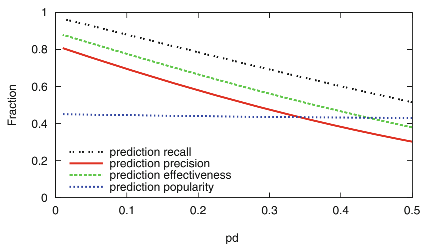
Figure 8a compares their recommendation performance by measuring the F1-Score values for various filter sizes. The x-axis represents the evolution of the probabilities pf, for our solution, and pd, for 2-DP. We show that the F1-Score of 2-DP decreases faster than ours. The F1-Score of 2-DP with a pd of at least 0.2 is smaller than that of our solution with a filter size greater than 100. In addition, revealing the most popular 10 % of the compact profile ($fs = 50$) yields similar performance as 2-DP with $pd \geq 0.3$.

Figure 8b measures the level of privacy as the overlap rate (computed with the Jaccard index) between the compact profile and the obfuscated profile: lower overlap rate implies more privacy. 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 an overlap rate of about 0.55 with an F1-Score (from Fig. 8a) of 0.55. Our approach, on the other hand yields the same overlap rate with a filter size between $150 < fs < 200$, which corresponds to an F1-Score value of about 0.57.

Figure 9, instead, assesses privacy by measuring if the items in a user's real profile can be predicted by an attacker that analyzes the user's public profile. Note that in 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 predictions. Prediction recall measures the fraction of correctly predicted items out of those in the compact profile. Prediction precision measures the fraction



(a) Prediction with OPRD, $pf = 0.2$



(b) Prediction with 2-DP

Fig. 9. Profile prediction

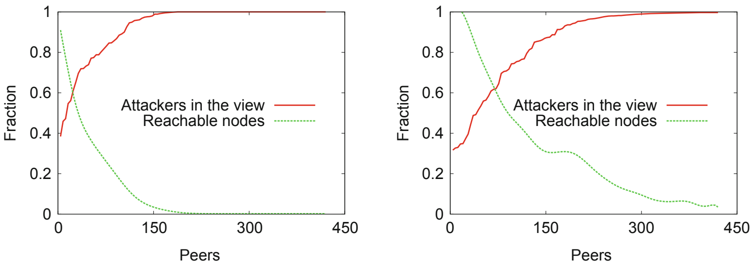
of correct predictions out of all the prediction attempts. For our solution, in Fig. 9a, we use a $pf = 0.2$ to control the randomized dissemination, and vary the filter size. For 2-DP (Fig. 9b), we instead vary pd.

The plots show that while our approach is subject to fairly precise predictions, these cover only a small fraction of the compact profile with reasonable values of fs. With $fs = 200$, the prediction recall is of about 30 %. In contrast, 2-DP exposes a higher number of items from the compact profile. With $pd = 0.2$ the prediction recall is 0.8 with a prediction precision of 0.6. The curves for prediction effectiveness, computed as F1-Score values, 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. We observe that when the filter size decreases, the correctly predicted items are among the most popular ones, which are arguably the least private.

6.6 Resilience to a Censorship Attack

We illustrate the resilience of our obfuscation protocol against censorship by implementing a simple eclipse attack [18]. 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 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. We evaluate the efficiency of this attack with two metrics: the poisoning rate of the target's clustering view by attackers; and the fraction of honest nodes (*e.g.* not censors) reachable by the target 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 10a and b 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



(a) 2-DP, $pd = 0.2$

(b) OPRD, $fs = 125$, $pf = 0.2$

Fig. 10. Resilience to censorship

150 nodes have a poisoning rate lower than 1. This is because 2-DP computes similarities using the randomized compact profile, which it also shares with other users. Therefore 2-DP exhibits exactly the same vulnerability as CT. The censors can trivially match the profile of the target node.

Our approach is more resilient to this censorship attack. It is difficult for censors to intercept all 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 node's profile. Consequently, their profiles are more likely to resemble those of users with correlated interests than to match the target profile. Figure 8b confirms this observation by showing the overlap between obfuscated and compact profiles. The resilience of OPRD is driven by the size of the obfuscation filter, the smaller the filter, the more resilient the protocol.

6.7 Bandwidth Consumption

We also conducted experiments using our prototype with 215 users running on approximately 110 PlanetLab nodes in order to evaluate the reduction of network cost resulting from the compactness of our profiles. The results in terms of F1-Score, recall, and precision closely mimic those obtained with our simulations and are therefore omitted. The bandwidth cost associated with our obfuscated solution (not depicted for space reason) is about one third of that of the solution based on cleartext profiles [5].

7 Related Work

Privacy is important in many applications. Several approaches [2,17] use randomized distortion techniques to preserve the privacy of sensitive data. However, [13,15] show that random distortion can seriously compromise privacy. Instead of adding randomness to user profiles, our solution uses coarse-grained profiles that reveal only the least sensitive information. The changes we apply to profiles are thus not random, but they depend on the interests of users. This makes it harder to separate privacy sensitive information from the introduced distortion.

Some authors [1] designed a statistical measure of privacy based on differential entropy. However, it is difficult to evaluate its meaning and its impact on sensitive data. Differential privacy was considered in [9,12]. In a distributed settings, [4] proposed a differentially private protocol to measure the similarity between peers. While this solution works well with 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, [16] highlights the trade-off between privacy and accuracy.

Other approaches [7] exploit homomorphic encryption in a P2P environment to secure multi-party computation techniques. Similarly, [3] proposes an architecture for privacy preserving CF by replacing the single server providing the service with a coalition of trusted servers.

8 Concluding Remarks

The motivation of this work is to make distributed CF resilient to privacy and censorship attacks without jeopardizing the quality of recommendation. We proposed a mechanism that relies on two components: (i) an obfuscation scheme revealing only the least sensitive information in the profiles of users, and (ii) a randomization-based dissemination protocol ensuring differential privacy during the dissemination. We showed the viability of our mechanism by comparing it with a non-private and a fully (differentially) private alternative. However, many questions remain open. In particular, evaluating the fundamental trade-offs between privacy, resilience to censorship, and recommendation quality constitutes an interesting research direction.

References

1. Agrawal, D., Aggarwal, C.C.: On the design and quantification of privacy preserving data mining algorithms. In: PODS (2001)
2. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: SIGMOD (2000)
3. Ahmad, W., Khokhar, A.: An architecture for privacy preserving collaborative filtering on web portals. In: IAS (2007)
4. Alaggan, M., Gambs, S., Kermarrec, A.-M.: BLIP: non-interactive differentially-private similarity computation on bloom filters. In: Richa, A.W., Scheideler, C. (eds.) SSS 2012. LNCS, vol. 7596, pp. 202–216. Springer, Heidelberg (2012)
5. Boutet, A., Frey, D., Guerraoui, R., Jégou, A., Kermarrec, A-.M.: Privacy-preserving distributed collaborative filtering. Technical report RR-8253, INRIA, March 2013
6. Boutet, A., Frey, D., Guerraoui, R., Jégou, A., Kermarrec, A.-M.: WhatsUp Decentralized Instant News Recommender. In: IPDPS (2013)
7. Canny, J.: Collaborative filtering with privacy via factor analysis. In: SIGIR (2002)
8. Das, A.S., Datar, M., Garg, A., Rajaram, S.: Google news personalization: scalable online collaborative filtering. In: WWW (2007)

9. Dwork, C.: Differential privacy: a survey of results. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008)
10. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
11. Goldreich, O.: Cryptography and cryptographic protocols. Distrib. Comput. **16**, 177–199 (2003)
12. Haeberlen, A., Pierce, B.C., Narayan, A.: Differential privacy under fire. In: SEC (2011)
13. Huang, Z., Du, W., Chen, B.: Deriving private information from randomized data. In: SIGMOD (2005)
14. Kanerva, P., Kristoferson, J., Holst, A.: Random indexing of text samples for latent semantic analysis. In: CCSS (2000)
15. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: On the privacy preserving properties of random data perturbation techniques. In: ICDM (2003)
16. Machanavajjhala, A., Korolova, A., Sarma,A.D.: Personalized social recommendations: accurate or private. In: VLDB (2011)
17. Polat, H., Du, W.: Svd-based collaborative filtering with privacy. In: SAC (2005)
18. Singh, A., Castro, M., Druschel, P., Rowstron, A.: Defending against eclipse attacks on overlay networks. In: SIGOPS (2004)
19. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Adv. Artif. Intell. (2009)
20. Tarkoma, S., Rothenberg, C.E., Lagerspetz, E.: Theory and practice of bloom filters for distributed systems. IEEE Commun. Surv. Tutorials **14**, 131–155 (2012)
21. van Rijsbergen, C.J.: Information Retrieval. Butterworth, London (1979)
22. Voulgaris, S., Gavidia, D., van Steen, M.: Cyclon: inexpensive membership management for unstructured p2p overlays. J. Netw. Syst. Manage. **13**, 197–217 (2005)
23. Voulgaris, S., van Steen, M.: Epidemic-style management of semantic overlays for content-based searching. In: Cunha, J.C., Medeiros, P.D. (eds.) Euro-Par 2005. LNCS, vol. 3648, pp. 1143–1152. Springer, Heidelberg (2005)
24. Wan, M., Jönsson, A., Wang, C., Li, L., Yang, Y.: A random indexing approach for web user clustering and web prefetching. In: Cao, L., Huang, J.Z., Bailey, J., Koh, Y.S., Luo, J. (eds.) PAKDD Workshops 2011. LNCS, vol. 7104, pp. 40–52. Springer, Heidelberg (2012)