Mediated Secure Multi-Party Protocols for Collaborative Filtering

EREZ SHMUELI, Tel-Aviv University, Israel
TAMIR TASSA, The Open University, Israel

Recommender systems have become extremely common in recent years, and are utilized in a variety of domains such as movies, music, news, products, restaurants, etc. While a typical recommender system bases its recommendations solely on users' preference data collected by the system itself, the quality of recommendations can significantly be improved if several recommender systems (or vendors) share their data. However, such data sharing poses significant privacy and security challenges, both to the vendors and the users. In this paper we propose secure protocols for distributed item-based Collaborative Filtering. Our protocols allow to compute both the predicted ratings of items and their predicted rankings, without compromising privacy nor predictions' accuracy. Unlike previous solutions in which the secure protocols are executed solely by the vendors, our protocols assume the existence of a mediator that performs intermediate computations on encrypted data supplied by the vendors. Such a mediated setting is advantageous over the non-mediated one since it enables each vendor to communicate solely with the mediator. This yields reduced communication costs and it allows each vendor to issue recommendations to its clients without being dependent on the availability and willingness of the other vendors to collaborate.

Additional Key Words and Phrases: Item-based collaborative filtering, distributed computing, Privacy

ACM Reference Format:
Erez Shmueli and Tamir Tassa. 2019. Mediated Secure Multi-Party Protocols for Collaborative Filtering. *ACM Trans. Intell. Syst. Technol.* 1, 1 (October 2019), 25 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

The explosion of information that is available to users over the World Wide Web was the main driving force in the emergence of recommendation systems that aim at helping users find their needles in the haystack [41]. One of the main techniques on which recommendation systems are based is collaborative filtering (CF) [17]. CF predicts the preferences of users based on the preferences of the community (rather than on the users' characteristics). Specifically, CF methods use large databases that store information regarding rating or purchasing history of users in the community.

CF is broadly classified into memory-based and model-based approaches. The memory-based approach is either user-based, in the sense that recommendations for a given user u are derived from the preferences of users that are similar to u, or item-based, i.e., u is offered items which are similar to those that he purchased or liked in the past. The item-based approach is more suitable for deployment in e-commerce sites, due to its better scalability, and, indeed, it is one of the most widely deployed CF techniques [13].

Authors' addresses: Erez Shmueli, Tel-Aviv University, Tel-Aviv, Israel, shmueli@tau.ac.il; Tamir Tassa, The Open University, Ra'anana, Israel, tamirta@openu.ac.il.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.

2157-6904/2019/10-ART $15.00
https://doi.org/0000001.0000001

Since memory-based approaches utilize all history of data, some of them do not scale well. In such cases, model-based approaches may be more applicable. In model-based approaches, the past rating or purchasing data is used to train a compact model, which is then used for prediction. The model is developed by artificial intelligence techniques (e.g., Bayesian classification) or by linear algebraic techniques (e.g., SVD). While item-based methods have comparable performance to that of model-based methods when predicted ratings are the required output, the former methods tend to show better performance when the required output is the top h items for a given user [13]. Furthermore, item-based recommendations are more easily interpretable to the user.

Instead of basing their recommendations to their clients solely on their own databases, vendors may significantly improve the quality of their recommendations by sharing their user-item preferential data [8]. However, such sharing poses significant privacy and security challenges. Indeed, commercial organizations may be reluctant to share their proprietary data about past users' purchases or ratings as it may serve competing entities. In addition, the users' privacy might be hindered if personal information about their past activities which they provided to one commercial entity would be handed over by that entity to other entities.

Privacy-preserving collaborative filtering (PPCF) enables the practice of CF without leaking private information. One class of PPCF algorithms is based on techniques such as data perturbation or generalization. The outputs issued by such algorithms may differ from the outputs of their non-privacy preserving counterparts, due to the noise that they introduce to the training data. Another class of PPCF algorithms is based on cryptographic techniques, such as homomorphic encryption or secret sharing. Employing cryptographic techniques, rather than data perturbation, offers better privacy-preservation and it enables issuing the same output as in the the underlying non-privacy preserving algorithm.

Here we offer Secure Multi-party Computation (SMC) protocols for item-based collaborative filtering in two distributed settings: the vertical distribution case, where each of the collaborating vendors (denoted $V_1, \ldots, V_K$) offers a different subset of items to the same underlying population of users, and the horizontal distribution setting, in which $V_1, \ldots, V_K$ offer the same set of items but each vendor serves a distinct subset of users. Our protocols in the vertical setting can be used, say, by a news website and a movie rental website in order to improve their recommendations (for either news items or movies) to their clients. The protocols in the horizontal setting can be used by several online bookstores that serve different groups of users (say, each bookstore serves a different geographical region). In both settings, our protocols are designed for computing predicted ratings as well as predicted rankings.

While almost all prior art on PPCF concentrated on the case of two vendors only ($K = 2$), as we detail in Section 2, a major contribution of our solutions is that they apply for any number of vendors. Devising a PPCF solution for a general number of vendors is of utmost importance, as the whole point of collaboration in the context of CF is to increase the training dataset towards inferring better recommendations. Hence, PPCF solutions that assume only two vendors may be insufficient in typical scenarios that require the conjoining of more private databases towards increasing the quality of recommendations. Alas, when dealing with a general number K of vendors, hefty theoretical and practical issues arise. We tackle those issues by adopting "the mediated model" [2, 3]. It is our belief that such a paradigm shift is an essential step towards making PPCF viable and essential in real-life recommendation systems.

The paper is organized as follows. We begin with a review of related work (Section 2). In Section 3 we discuss the mediated model, and the reasons that we consider it to be a most suitable model for performing PPCF with a general number K of collaborating vendors. We then provide the necessary background on item-based CF, distributed scenarios, PPCF and cryptographic methods (Section 4). In Sections 5 and 6 we describe our PPCF protocols in the vertical and horizontal distribution scenarios, respectively. We present our experimental evaluation in Section 7, and conclude in Section 8.

A preliminary version of this work appeared in the 11th ACM Conference on Recommender Systems [43].

2 RELATED WORK

Privacy-preserving data mining (PPDM) [1, 28] is the practice of performing data mining on private data which is distributed among several parties. The goal in such cases is to jointly perform data mining on the unified corpus of data, while protecting the data records of each of the data owners from its peers. Application examples include e.g. construction of decision trees [28], clustering [14, 22, 27], association rule mining [24, 44, 53, 57], classification [10, 29, 30, 50], computation on distributed graphs [4, 5, 7], and distributed constraint optimization problems [18–20, 25, 48, 51, 52].

Privacy-preserving collaborative filtering (PPCF) falls under the umbrella of PPDM. It enables the practice of CF without leaking private information. We proceed to overview several recent PPCF works. Interested readers may refer to [9] and [15] for a more comprehensive survey.

Polat and Du [36] discuss how to provide predictions for single items in case of a vertical distribution scenario with two vendors. They take the user-based approach: for each pair of users they compute a cosine-like similarity score and then they predict the rating that a user u would give to item b based on a weighted average of all ratings that were issued for b by any user, where the users are weighted by their similarity to u.

In another study of theirs, Polat and Du [37] show how to offer top recommendations in both the horizontal and vertical distribution scenarios without deeply violating the vendors' privacy. They concentrate on the case of two vendors, and the case of binary user-item data (namely, did user u like or not item b). Here too, the approach is user-based, i.e., the recommendations to u are derived from the items that users similar to u liked.

Jeckmans et al. [21] consider a case of horizontal distribution between two vendors, V_1 and V_2. Their CF approach is also user-based: they first build a model of similarity between users, and then use user neighborhoods in order to compute predicted ratings. They consider an asymmetric setting in which V_2 collaborates with V_1 so that V_1 can offer to its clients better recommendations, while V_2 does not benefit at all from such a collaboration.

Basu et al. [6] propose algorithms for a privacy-preserving execution of an item-based CF scheme that is based on the Slope One predictor [26]. Their scheme uses additively homomorphic public-key encryptions. Their algorithms consist of an offline pre-computation phase and an online prediction phase. In the online phase, whenever one vendor needs to issue a predicted rating, all parties collaborate in order to recover the predicted rating. Their timing information, in the order of seconds, is based on a prediction for a single user and single book. This is after pre-computation in the order of hours.

The work which is closest to ours is that of Yakut and Polat [55]. To the best of our knowledge, it is the only other work that deals with item-based CF which is based on the cosine-similarity score. They concentrate on the case of two vendors and considered the arbitrary distribution scenario. A main vehicle by which they obtain privacy is by introducing fake ratings into the distributed matrix of user-item ratings. Hence, as opposed to our algorithms that issue exactly the same outputs as their non-privacy preserving counterparts, the outputs of the algorithms by Yakut and Polat [55] are inaccurate, and in order to enhance the privacy, greater levels of noise must be introduced, what implies a further reduction of accuracy.

The works which we proceed to describe differ from the works described above and from the present study as they relate to a non-distributed setting, in which a central entity holds the entire user-item matrix. Those works utilize randomization techniques in order to prevent the server from learning private rating information of users, as well as to prevent users from using the predicted ratings that are presented to them by the CF algorithm in order to infer information on neighboring users.

The studies of Polat and Du [34, 35] and Polatidis et al. [38] utilize random perturbations on users' ratings before they are submitted to the central server. They consider CF algorithms that are user-based. In [16], which also considers user-based CF, the perturbation is applied on the similarity scores between users, rather than on the entries of the user-item matrix. The study of Zhu et al. [59] differs from the above-mentioned studies, as the added random noise is calibrated towards achieving a certain level of differential privacy [12]. Their CF algorithm is item-based. In fact, the differential privacy techniques that are presented in [59] are complimentary to ours and can be implemented on top of the cryptographic methods that we present herein.

3 THE ADVANTAGES OF MEDIATION FOR PPCF

All of the studies that we reviewed in Section 2, except the work of Basu et al. [6], concentrate on the case of $K = 2$ vendors. However, in order to perform a more effective CF, larger values of K may be required. A natural extension of the schemes in those studies to a general K (in order to enable better recommendations) implies that whenever a vendor wishes to compute predicted ratings or rankings, all other vendors need to be available and participate in the computation. Such constant involvement of the vendors is problematic for several reasons. First, those schemes assume that every pair of vendors must have a permanent channel of communication; such an assumption may be impractical, and it also entails $O(K^2)$ communication costs. Second, if one of the vendors' servers is down, other vendors may be prevented from issuing recommendations. Third, different vendors have different recommendation practices, and in the horizontal setting different vendors may have clienteles of different sizes; such diversities create an imbalance between the vendors' contribution to the general workload. Also the work of Basu et al. [6], which is the only work so far that offered a PPCF solution for a general number of vendors, suffers from all three problems that we described above. (They reduce the communication costs from $O(K^2)$ to $O(K)$ by assuming a circular network that connects all vendors; alas, such an assumption is also impractical, because it still requires permanent peer-to-peer communication channels between vendors.)

A simple way to address all those issues and allow meaningful PPCF (namely, PPCF that is not limited to $K = 2$ vendors only) is to adopt "the mediated model" [2, 3]. In that model, it is assumed that the agents may be assisted by a third party (or third parties) to whom they may export some computations. That third party, called *a mediator*, is trusted to perform the computations honestly, but at the same time it is not allowed access to private inputs of the agents. The protocols we suggest in this paper involve a mediator, T. Having such a mediator reduces the need of the vendors to communicate with each other only to the offline (and infrequent) phase in which a model of similarity between items is built; in the online phase, where vendors need to issue predicted ratings and rankings to their clients, each vendor communicates directly vis-à-vis T and does not need to communicate with other vendors. Such a mediated setting is advantageous over the non-mediated setting since it entails reduced run-time and communication costs, and each vendor can issue recommendations to its clients without "bothering" all other vendors each time or be dependent on their availability and willingness to collaborate. Specifically, if V_1 took one day off for whatever reason, it does not prevent V_2 from communicating with T and keep issuing recommendations to its clients; and if V_3 wishes to issue many recommendations, it may work vis-à-vis T as much as it needs (and pay T accordingly) without affecting V_4 who may issue recommendations much less frequently.

4 PRELIMINARIES

We provide here the necessary background about item-based CF (Section 4.1), distributed scenarios (Section 4.2), PPCF (Section 4.3), and relevant cryptographic building blocks (Section 4.4). Hereinafter, we use the following notation agreements:

(1) Matrices are denoted by capital letters, e.g. A, and their entries are denoted by $A(\cdot,\cdot)$. Vectors are denoted by bold face letters, e.g. $\mathbf{r}$, and their entries are denoted by $\mathbf{r}(\cdot)$.
(2) For a positive integer M, $[M] := \{1, 2, \ldots, M\}$.
(3) If r is a nonnegative integer then $\xi(r) = 0$ if $r = 0$ and $\xi(r) = 1$ otherwise.
(4) If $\mathbf{r}$ is a vector and f is any scalar function over the reals, then $f(\mathbf{r})$ will denote the vector in which $f(\mathbf{r})(\cdot) = f(\mathbf{r}(\cdot))$; namely, if $\mathbf{r} = (\mathbf{r}(1), \cdots, \mathbf{r}(M))$ then $f(\mathbf{r}) = (f(\mathbf{r}(1)), \cdots, f(\mathbf{r}(M)))$.
(5) Inner products between vectors will be denoted by $\langle\cdot,\cdot\rangle$; the induced norm will be denoted by $\|\cdot\|$.
(6) If $\mathbf{x}$ and $\mathbf{y}$ are two N-dimensional vectors then $\mathbf{xy}$ is the vector in which $(\mathbf{xy})(n) = \mathbf{x}(n)\cdot\mathbf{y}(n)$, $n \in [N]$, and $\mathbf{x}^{\mathbf{y}}$ denotes the scalar $\prod_{n=1}^{N} \mathbf{x}(n)^{\mathbf{y}(n)}$.

4.1 Item-based collaborative filtering

Here we provide a brief introduction to item-based CF [40]. A more comprehensive discussion is given in [13]. In what follows we introduce detailed mathematical notations which are not standard in prior art on CF. The reason for doing so is that some details of the recommendation algorithms are described in prior art only verbally, without using mathematical notations. We decided to formalize those details by introducing relevant mathematical notations, since such notations enable us, later on, to describe our PPCF protocols in a succinct and accurate manner.

Let $U = \{u_1, \ldots, u_N\}$ be a set of users and $B = \{b_1, \ldots, b_M\}$ be a set of items (say, books). The user-item rating matrix, R, is an $N \times M$ matrix where $R(n, m)$ is the rating that u_n gave to b_m, a value which is usually taken from a small range of positive integers, say $\{1, 2, 3, 4, 5\}$, and $R(n, m) = 0$ if u_n did not rate b_m. Item-based CF consists of two phases: an offline phase, in which the matrix R is used to learn a model of similarity between items; and an online phase in which that model is used to predict user ratings or to rank items according to their potential appeal to a given user. (The offline phase should be repeated periodically in order to update the similarity scores according to the changes in U, B, and R.)

The similarity model is an $M \times M$ symmetric matrix S where $S(\ell, m)$ is the similarity score between items b_ℓ and b_m, $\ell, m \in [M] := \{1, 2, \ldots, M\}$. Let $\mathbf{c}_m = (R(n, m) : n \in [N])$ denote hereinafter the mth column in the user-item rating matrix R, $m \in [M]$. Then the similarity scores are defined in Definition 4.1.

Definition 4.1. *Assume that $\ell, m \in [M]$ and set $\mathbf{c}_{\ell|m} := \mathbf{c}_\ell \cdot \xi(\mathbf{c}_m)$; namely, $\mathbf{c}_{\ell|m} := (\mathbf{c}_{\ell|m}(1), \ldots, \mathbf{c}_{\ell|m}(N))^t$ is the projection of the ℓth column of the user-item rating matrix R on the subset of users that rated both items b_ℓ and b_m. Then the cosine similarity score is*

$$S(\ell, m) = \frac{\langle \mathbf{c}_\ell, \mathbf{c}_m \rangle}{\|\mathbf{c}_{\ell|m}\| \cdot \|\mathbf{c}_{m|\ell}\|}\,, \tag{1}$$

where if $\mathbf{c}_{\ell|m} = \mathbf{0}$ or $\mathbf{c}_{m|\ell} = \mathbf{0}$, $S(\ell, m)$ is set to zero.

The similarity scores are used to predict u_n's rating of b_m as follows. Let:

- $q < M$ be a preset (typically small) integer.
- $N_q(m)$ be the set of indices of the q nearest neighbors of b_m (those with highest $S(\cdot, m)$).
- $N_q^+(m) := \{\ell \in N_q(m) : S(\ell, m) > 0\}$.
- $\mathbf{s}_m$ be the M-dimensional vector for which $\mathbf{s}_m(\ell) = S(\ell, m)$ if $\ell \in N_q^+(m)$ and $\mathbf{s}_m(\ell) = 0$ otherwise.

- $\overline{R(b_m)} = \left[\sum_{n\in[N]} R(n,m)\right] / \left[\sum_{n\in[N]} \xi(R(n,m))\right]$ be the average rating given to item b_m.
- $\mathbf{r}_n$ be the nth row of the user-item rating matrix R.
- $\bar{\mathbf{r}}_n$ be the vector of u_n's adjusted ratings, i.e. $\bar{\mathbf{r}}_n(m) = (R(n,m) - \overline{R(b_m)}) \cdot \xi(R(n,m))$, $m \in [M]$.

Then the predicted rating $P(u_n, b_m)$ is the weighted average over the adjusted ratings that u_n made thus far,

$$P(u_n, b_m) := \overline{R(b_m)} + \frac{\langle \mathbf{s}_m, \bar{\mathbf{r}}_n \rangle}{\langle \mathbf{s}_m, \xi(\mathbf{r}_n) \rangle} \,. \tag{2}$$

The weighted average is taken over at most q items, and it is based only on items that have a positive similarity to b_m. The quotient in Eq. (2) is undefined if the denominator equals zero (i.e., if none of the items that u_n had rated in the past is in $N_q^+(m)$). In that case, $P(u_n, b_m)$ is set to $\overline{R(b_m)}$. (There exist several versions of the similarity score and prediction formula. We focus here on the version that is suggested in [13] and, by our experiments, performs best. The adjustment of our protocols to other variants is straightforward.)

Sometimes, instead of showing to u_n his predicted rating on some item, the goal is to present to him the h items which are most likely to appeal to him, without predicted ratings. To that end, one produces a *ranking* of all items that u_n had not rated so far in order to extract from it the top h items, for some $h \geq 1$. One approach is to base the ranking on the items' predicted ratings (Eq. (2)). However, a simpler ranking procedure produces better results [13, Section 2.3]. Let $I(n)$ be the subset of indices of items that u_n already rated. Define for each $m \in [M] \setminus I(n)$ the score

$$\hat{s}(m) = \sum_{\ell \in I(n) \cap N_q(m)} S(\ell, m) \,. \tag{3}$$

Namely, $\hat{s}(m)$ is the sum of similarities between b_m and all those items that u_n already rated and fall within $N_q(m)$, the q-neighborhood of b_m, for some selection of q. Then, the top h items to be recommended to u_n are those with the highest value of $\hat{s}(m)$.

4.2 Distributed scenarios

When the entire corpus of data, namely the user-item rating matrix R, is held by one party, then that party can compute on its own predictions for future ratings or item rankings. However, it is beneficial for different parties (or *vendors*, as we shall refer to them hereinafter) to collaborate and conjoin their databases in order to better understand the characteristics of items and the likes of users and, consequently, issue better predictions. Therefore, in such settings, the computation of the similarity model in the offline phase as well as the computation of predictions in the online phase must be carried out over a database that is distributed in several sites.

We consider here scenarios in which the user-item rating matrix R is distributed among K vendors $V_1, \ldots, V_K$. The two main distributed scenarios are the following:

Vertical: All vendors serve all users in U but they offer disjoint subsets of items from B. Hence, each vendor owns a different subset of R's columns.

Horizontal: All vendors offer all items in B but they serve disjoint subsets of users from U. Hence, each vendor owns a different subset of R's rows.

The distributed settings raise the issue of privacy, which we discuss in the next section.

4.3 Privacy preserving collaborative filtering

In the general setting of SMC (Secure Multi-party Computation) [56], n mutually distrustful parties, $P_1, \ldots, P_n$, that hold private inputs, $x_1, \ldots, x_n$, wish to compute some joint function on their inputs, i.e., $y = f(x_1, \ldots, x_n)$, without revealing any unnecessary information about these inputs, except for what is logically learned from the output. For example, in case where n parties wish to compute the average of their salaries, the result reveals to each party the average of the other $n-1$ parties; the latter value is not part of the desired output, but it can be inferred by each party given the computed output and its own input. It is also possible that each party P_i has its own desired output, $y_i = f_i(x_1, \ldots, x_n)$, where f_i is a publicly known function. SMC enables the parties to compute the desired output function (or functions) using an interactive protocol. An SMC protocol is said to achieve *perfect* privacy, if each party P_i learns exactly y (or y_i), and whatever can be inferred from that output and its own input, but it learns nothing else.

In our setting, each vendor wishes to prevent the other parties from inferring information on the user-item rating information that it holds, because of commercial considerations and the need to respect its users' privacy. Hence, it is needed to perform the CF computations in a privacy-preserving manner.

Ideally, no party (any of the vendors, or the mediator) should gain any information on the user-item rating matrix R from its view during the execution of our protocols, beyond what can be inferred from its own input and output. Theoretical results show that such *perfect* privacy is achievable for any problem of SMC (by invoking generic solutions such as Yao's garbled circuit construction [56]). However, when looking for *practical* solutions, some relaxations of the notion of perfect privacy are usually inevitable, provided that the leaked information is deemed benign. Examples for such studies are numerous and span various domains of distributed computing, e.g. distributed association rule mining [24, 42, 44, 53, 57], anonymization of distributed datasets [23, 46, 49, 58], distributed constraint optimization problems [18–20, 25, 47, 48, 51, 52], distributed graph mining [5] and more. In fact, also all PPCF studies that were reviewed in Section 2 offer protection to the private data, but that protection is not perfect, in the sense that the protocols presented there do leak to each participating vendor some information on inputs of other vendors, beyond just the desired output of the computation.

The protocols that we present here are also privacy-preserving in that sense; namely, they protect the private data of each vendor, but not perfectly. For each of our protocols we discuss its privacy-preservation and identify the excess information that they may leak, and explain why such leakage of information is benign.

Our protocols rely solely on standard cryptographic building blocks, as described in Section 4.4; this fact offers a significant advantage as they rely on well-founded cryptographic building blocks and they can be readily implemented on top of standard libraries.

We make here an assumption which is common in PPCF literature: all parties, $V_1, \ldots, V_K$ and T, are semi-honest and do not collude. Semi-honesty means that the parties follow the protocols' specifications but try to extract from their view information on the inputs of their peers. The assumption of semi-honesty is very common in literature on secure multi-party protocols and privacy-preserving distributed computations.

Parties that deviate from the protocol's specifications (in order to gain more information on other parties' inputs, or in order to prevent other parties from getting the correct output) are called in the cryptographic literature *malicious*. Safeguarding against malicious parties results in less practical protocols; because of that, and because the assumption of correctly following the prescribed protocol is justified in practical deployments, the bulk of studies that deal with multi-party computations, especially those that focus on specific real-life applications, assume semi-honest parties. (In particular, all studies mentioned above make that assumption.)

4.4 Cryptographic building blocks

In this section we describe the cryptographic tools that will be used by our privacy-preserving CF protocols.

Homomorphic encryption. A cipher is called public-key if its encryption function $\mathcal{F}(\cdot)$ depends on one key, $\mathbf{k}_e$, which is publicly known, while the corresponding decryption function $\mathcal{F}^{-1}(\cdot)$ depends on a private key $\mathbf{k}_d$ that is known only to the owner of the cipher, and $\mathbf{k}_d$'s derivation from $\mathbf{k}_e$ is computationally hard.

A public-key cipher is called (additively) *homomorphic* if its plaintext domain is an additive commutative group, its ciphertext domain is a multiplicative commutative group, and for every two plaintexts, x_1 and x_2, $\mathcal{F}(x_1 + x_2) = \mathcal{F}(x_1) \cdot \mathcal{F}(x_2)$.

When the encryption function is randomized (in the sense that $\mathcal{F}(x)$ depends on x as well as on a random string), $\mathcal{F}$ is called *probabilistic.* Hence, a probabilistic encryption function is a one-to-many mapping (every plaintext x has many encryptions $y = \mathcal{F}(x)$), while the corresponding decryption function is a many-to-one mapping (all possible encryptions y of the same plaintext x are mapped by $\mathcal{F}^{-1}(\cdot)$ to the same x).

The semantically secure Paillier cipher [33] is a public-key cipher that is both homomorphic and probabilistic. Its plaintext and ciphertext domains are $\mathbb{Z}_\nu$ and $\mathbb{Z}^*_{\nu^2}$, respectively, where the so-called modulus ν is a product of two large primes, p and q.

In our protocols, $V_1, \ldots, V_K$ jointly generate the key pair in a homomorphic and probabilistic encryption function $\mathcal{F}$; they notify T of the encryption key, but keep the decryption key private. (In order to jointly generate a random key pair, the vendors only need to generate a random bit string of the same length as the key; this can be done, say, if every vendor chooses its own random string and then they all engage in a secure summation protocol.)

The encryption function needs to be homomorphic in order to enable arithmetic computations in the ciphertext domain. It needs to be probabilistic as that property is essential when dealing with plaintexts that come from a small and publicly known domain (such as binary plaintexts or rating values, as explained in Section 4.1). Indeed, if we had utilized a deterministic encryption function then an adversary would have been able to detect occurrences of equal plaintexts and even deduce some of the plaintexts. A probabilistic encryption function, on the other hand, prevents such inferences since it creates each time a new and random-dependent ciphertext.

Secure division. Another building block of secure multi-party computation that we shall need is that of secure division. Tassa and Bonchi [45] considered a setting that involves three parties, P_1, P_2 and H. P_i, $i = 1, 2$, holds a private integer a_i, and the goal is to let H recover the real quotient a_1/a_2 but not the values of a_1 and a_2. Towards that end, P_1 and P_2 jointly generate a random real number $X \sim Z$ where Z is the distribution on $[1, \infty)$ with probability density function $f_Z(x) = x^{-2}$; then they jointly generate a random $g \sim U(0, X)$, and they send to H the values ga_i, $i = 1, 2$, which H proceeds to divide in order to recover $q = a_1/a_2$. The masking multiplier g prevents H from learning a_i, but it can still recover q. The selection of the probability distribution from which g is drawn is made in order to minimize the information that H can extract from ga_i on a_i. Specifically, the a-posteriori belief probability of H regarding the value of a_i after observing ga_i is close to whatever a-priori belief probability H had regarding a_i prior to seeing ga_i. (The reader is referred to [45] for the full analysis and experimentation of this approach.) Here we use this idea as follows: T holds two encrypted values $\mathcal{F}(a_1)$ and $\mathcal{F}(a_2)$ and it wishes to allow a vendor V_k to get the quotient $q = a_1/a_2$ so that no one reveals a_1 nor a_2. To that end it sends to V_k the values $b_i := \mathcal{F}(a_i)^g$, $i = 1, 2$, for a random g that was generated as described above. Since $\mathcal{F}$ is homomorphic then $b_i := \mathcal{F}(ga_i)$. V_k decrypts the two received values and gets $\mathcal{F}^{-1}(b_i) = ga_i$, $i = 1, 2$, which it then proceeds to divide in order to get q.

5 PRIVACY PRESERVING ALGORITHMS FOR THE SCENARIO OF VERTICAL DISTRIBUTION

Here we deal with the vertical distribution scenario, where each vendor owns a different subset of R's columns. In Sections 5.1 and 5.2 we describe the computations and protocols that are performed in the offline phase and involve all of the vendors $V_1, \ldots, V_K$, as well as the mediator T. Then, we describe the online phase in which a given vendor V_k submits queries to T towards computing the predicted rating of some user u_n for an item b_m, $P(u_n, b_m)$, Eq. (2) (Section 5.3), or getting the top h items for a given user u_n (Section 5.4). The online phase is carried out solely by V_k and T. Namely, the participation of all vendors is required only in the offline and less frequent phase.

User ordering. The vendors jointly decide on a random ordering of U, which is kept secret from T. Namely, even if T knows the ground-set of users, U, it will not know who is the user that corresponds to a given index $n \in [N]$. Hereinafter, u_n denotes the nth user in that ordering.

Item ordering. The vendors jointly decide on a random ordering of the unified item set B. Let $I_k \subset [M]$ denote the subset of indices of V_k's items in that ordering, $k \in [K]$. T is notified of I_k, $k \in [K]$, but apart from that the ordering is kept secret from T. If $M_k := |I_k|$ is the number of items offered by V_k, $k \in [K]$, then for every $m \in I_k$, the vector $\mathbf{c}_m$ (the mth column in the user-item rating matrix R) is known only to V_k.

We note that in order to select those random orderings, all that is needed is to select a common random seed; such a common random seed can be generated as described in Section 4.4.

5.1 Offline model construction

The split $[M] = \bigcup_{k=1}^{K} I_k$ induces a split of the similarity matrix S into K^2 blocks, $S_{j,k}$, $j, k \in [K]$, where the dimensions of the block $S_{j,k}$ are $M_j \times M_k$. The K diagonal blocks, $S_{k,k}$, $k \in [K]$, consist of similarity scores between pairs of items that are offered by the same vendor. Each V_k can compute by itself the block $S_{k,k}$ and send it to T. The computation of $S_{j,k}$, where $j \neq k$, depends on inputs from V_j and V_k. Protocol 1 enables that computation.

Assume that b_ℓ is one of the items offered by V_j and b_m is one of V_k's items. Protocol 1 is executed by V_j, V_k and T towards the goal of T learning $S(\ell, m)$. The protocol relies on a sub-protocol SSP for Secure Scalar Product. If $\mathbf{x}$ is a vector held by V_j and $\mathbf{y}$ is a vector held by V_k, then an execution of SSP with those two inputs ends with T learning $\langle \mathbf{x}, \mathbf{y} \rangle$, and nothing further, while both V_j and V_k remain oblivious of the input vector of the other vendor.

First, V_j (where $j < k$) selects a random multiplier $g_{\ell,m}$ (Step 1). Then they execute SSP towards T receiving $z_1 = g_{\ell,m} \cdot \langle \mathbf{c}_\ell, \mathbf{c}_m \rangle$ (Step 2). In Steps 3 and 4 T gets $z_2 = g_{\ell,m} \cdot \langle \mathbf{c}_\ell^2, \xi(\mathbf{c}_m) \rangle$ and $z_3 = g_{\ell,m} \cdot \langle \xi(\mathbf{c}_\ell), \mathbf{c}_m^2 \rangle$. Since $z_2 = g_{\ell,m} \cdot \|\mathbf{c}_{\ell|m}\|^2$ and $z_3 = g_{\ell,m} \cdot \|\mathbf{c}_{m|\ell}\|^2$, we infer by Eq. (1) that in Step 5 T recovers $S(\ell, m)$. (Recall that, as discussed in Section 4.1, if the denominator in that quotient is zero, a case that occurs when no two users rated both items, the similarity score is set to zero.)

Secure computation of scalar products of private vectors is a fundamental problem in SMC, as it serves a basic building block for many other secure protocols (see e.g. [54] and the references therein). Protocol 1 can be executed with any SSP sub-protocol. We find the protocol of Du and Zhan [11] most fitting in our context, since their protocol is designed for the mediated setting which we consider; namely, two parties hold each of the input vectors and the scalar product goes to a mediator. In addition, because it relies on a mediator, it solves the problem with perfect security without resorting to expensive cryptographic means, thus implying low computational and communication costs.

Protocol 1 Computing the similarity score for a single pair of items.

```
Require: $V_j$ holds the vector $\mathbf{c}_\ell$ for item $b_\ell$;
    $V_k$ holds the vector $\mathbf{c}_m$ for item $b_m$.
1: $V_j$ selects a random integer multiplier $g_{\ell,m}$.
2: $T \leftarrow z_1 := SSP(g_{\ell,m} \cdot \mathbf{c}_\ell, \mathbf{c}_m)$.
3: $T \leftarrow z_2 := SSP(g_{\ell,m} \cdot \mathbf{c}_\ell^2, \xi(\mathbf{c}_m))$.
4: $T \leftarrow z_3 := SSP(g_{\ell,m} \cdot \xi(\mathbf{c}_\ell), \mathbf{c}_m^2)$.
5: $T$ sets $S(\ell, m) := z_1/\sqrt{z_2 z_3}$ if $z_2 z_3 \neq 0$, and $S(\ell, m) = 0$ otherwise.
Ensure: $T$ gets $S(\ell, m)$.
```

Protocol 1 ends with T having the similarity score $S(\ell, m)$ between a single pair of items offered by V_j and V_k. V_j and V_k have to perform Protocol 1 in parallel for all $\ell \in I_j$ and $m \in I_k$. That parallelized version of Protocol 1 has to be run by all $\binom{K}{2}$ pairs of vendors.

In case items b_ℓ and b_m were never rated by the same user, then the scalar product that is computed in Step 2 of Protocol 1 (which is the numerator in Eq. (1), multiplied by $g_{\ell,m}$) will be zero. That is a welcome result, since the similarity score of two items that were never rated by the same user should indeed be set to zero. However, if those two items were both rated only by a single user, the similarity score will be maximal. To avoid such a potentially misleading similarity score, it is favorable to apply Eq. (1) only when the two items in question were rated by at least some threshold number of users. It is possible to enhance Protocol 1 to incorporate such a modification, by invoking secure set intersection computations. We omit further details.

Privacy. By using a secure SSP sub-protocol (such as the one in [11]), none of the vendors may infer any information on inputs of other vendors. As for T, the usage of random multipliers prevents it from learning the terms in the numerator and denominator of Eq. (1). T does learn the similarity scores between items, but, owing to the random item ordering, it cannot link those scores to the relevant pair of items. More formally, assume that T knows the exact item set B, and let S be the $M \times M$ matrix of similarity scores between them (see Section 4.1). Then at the conclusion of Protocol 1 T learns $P^t SP$ where P is the matrix corresponding to the random ordering of the item set B. Since P is selected uniformly at random and is kept secret from T, T cannot learn the similarity scores between any two specific items. But such a permuted matrix does leak to T excess information on the items such as the distribution of item similarity scores, existence of outlier items (items that have similarity scores smaller than some threshold σ with more than N_1 items, for some preset threshold $N_1 < N$), and so forth. Such excess information, however, is of benign nature since privacy is a notion that is more relevant to the human users than to the items themselves, and it seems reasonable that such similarities between items can be, most of the time, estimated quite well just from the features of the items, without seeing private rating data.

Computational and communication costs. In analyzing the computational costs of each of our protocols we focus only on the time consuming operations of encryption, decryption and exponentiation. (In what follows we denote the costs of those operations by $[Enc]$, $[Dec]$ and $[Exp]$, respectively.) Since the SSP protocol of Du and Zhan [11] is based only on operations like additions, multiplications and random number generations, that have significantly smaller runtimes, the computational cost of Protocol 1 (which is an offline protocol) is small. As for the communication cost, it involves, in total, four communication rounds, with an overall exchange of $O(M^2 N)$ bits.

A concluding remark. The similarity scores $S(\ell, m)$ are real-valued and in the next phase they need to be used in computing predicted ratings and rankings in a privacy-preserving manner. In doing so, it is necessary to subject them to encryption. Since encryption functions are applied on discrete integer domains, T translates all real values s in the matrix S into integer values by mappings of the form $s \mapsto \hat{s} := \lfloor Ls + 0.5 \rfloor$, where L is a large integer. Since the plaintext domains of public-key encryption functions are very large (at least $O(2^{512})$), there is no problem to choose a sufficiently large L so that $\frac{\hat{s}}{s} \approx L$. Such mappings will have a negligible effect on both rating and ranking computations. Indeed, when computing predicted ratings, both the numerator and denominator in the quotient on the right hand side of Eq. (2) are (approximately) multiplied by the same factor L. (It is true that the predicted rating based on the original similarity scores will differ from that which is based on the rescaled and rounded similarity scores, but if L is chosen sufficiently large, the difference becomes negligible). Likewise when computing predicted ranking based on the scores in Eq. (3). In what follows, $S(\ell, m)$ denote the rescaled and rounded integral similarity scores.

5.2 In preparation to the online phase

The computation of predicted ratings depends on $\overline{\mathbf{r}}_n$ and $\xi(\mathbf{r}_n)$, see Eq. (2). The vectors $\overline{\mathbf{r}}_n$ are real-valued. We translate them into integer-valued vectors $\hat{\mathbf{r}}_n$, where $\hat{\mathbf{r}}_n(m) := \lfloor L\overline{\mathbf{r}}_n(m) + 0.5 \rfloor$, and L is a large integer as described in the concluding remark of Section 5.1. Then, in the online phase, the predicted ratings will be computed as follows:

$$P(u_n, b_m) := \overline{R(b_m)} + \frac{1}{L} \cdot \frac{\langle \mathbf{s}_m, \hat{\mathbf{r}}_n \rangle}{\langle \mathbf{s}_m, \xi(\mathbf{r}_n) \rangle} \,. \tag{4}$$

Up to negligible rounding errors, Eq. (4) issues the same predictions as Eq. (2), while relying only on integer values.

Protocol 2 is designed so that at its completion T holds an $\mathcal{F}$-encryption of the vectors $\hat{\mathbf{r}}_n$ and $\xi(\mathbf{r}_n)$, for all $n \in [N]$, where $\mathcal{F}$ is the homomorphic encryption function that can be decrypted by the vendors only. Owing to the security of the $\mathcal{F}$-encryption, this protocol keeps the information owned by the vendors protected from T.

Protocol 2 Conveying to T encryptions of user-rating vectors.

```
Require: Each $V_k$, $k \in [K]$, holds $R(n,m)$ for all $n \in [N]$ and $m \in I_k$.
Each $V_k$ computes for each $m \in I_k$ the average rating of $b_m$, $\overline{R(b_m)} = \frac{\sum_{n\in[N]} R(n,m)}{\sum_{n\in[N]} \xi(R(n,m))}$.
Each $V_k$ computes the adjusted user-item matrix over its items, $\overline{R(n,m)} := \left(R(n,m) - \overline{R(b_m)}\right) \cdot \xi(R(n,m))$, $n \in [N], m \in I_k$.
Each $V_k$ computes $\hat{R}(n,m) = \lfloor L\overline{R(n,m)} + 0.5 \rfloor$, $n \in [N], m \in I_k$.
Each $V_k$ sends to $T$ the matrices $(\mathcal{F}(\hat{R}(n,m)) : n \in [N], m \in I_k)$ and $(\mathcal{F}(\xi(R(n,m))) : n \in [N], m \in I_k)$.
$T$ concatenates the $K$ received matrices.
Ensure: $T$ gets $\mathcal{F}(\hat{\mathbf{r}}_n)$ and $\mathcal{F}(\xi(\mathbf{r}_n))$, $\forall n \in [N]$.
```

Privacy. In Protocol 2 only the mediator T gets information. That information is encrypted by $\mathcal{F}$, and therefore it is protected from T and can only be used, by applying homomorphic arithmetic later on, to produce other $\mathcal{F}$-encrypted values that will be sent back to the vendors. No party (be it a vendor, the mediator, or an eavesdropper) receives any part of the user-item matrix R which it does not own.

Note that practical deployments of our protocols should be enhanced with standard security mechanisms. In particular, each party should create its own pair of private and public keys, and get from a Certificate Authority a corresponding certificate. Then, each message must be signed by the sender and encrypted under the receiver's public key. (The cost of such external encryption is negligible with respect to the encryptions in Protocol 2 since the body of each message can be encrypted using a symmetric cipher, the cost of which is dramatically smaller than that of public key ciphers, and then only the symmetric key needs to be encrypted using the receiver's public key.) Such standard security layers are essential when implementing *any* SMC protocol, and they come *on top* of the SMC protocol. They are essential in order to prevent eavesdropping, masquerading, and other well-known attacks on communication systems. In this study we focus only on the SMC protocols and not on the standard security mechanisms that serve as an external layer of protection.

Computational and communication costs. The computational cost, due to Protocol 2, for V_k is $2NM_k[Enc]$ (Step 4). The protocol includes one communication round with K messages of total length of $2NM\lambda$ bits (where λ denotes the number of bits in encrypted values).

5.3 Computing predicted ratings

In this phase, any vendor V_k, $k \in [K]$, can submit a query to T for the predicted rating of a single user $u \in U$ of an item b offered by V_k. Such queries are answered, in a privacy preserving manner, by Protocol 3. In the protocol we rely upon the following lemma.

Lemma 5.1. *Let $\mathcal{F}$ be a homomorphic encryption function and let $\mathbf{x}$ and $\mathbf{y}$ be two N-dimensional integer vectors. Denote $(\mathcal{F}(\mathbf{x}))^{\mathbf{y}} := \prod_{n=1}^{N} \mathcal{F}(\mathbf{x}(n))^{\mathbf{y}(n)}$. Then $(\mathcal{F}(\mathbf{x}))^{\mathbf{y}} = \mathcal{F}(\langle \mathbf{x}, \mathbf{y} \rangle)$.*

Proof. The homomorphic property of $\mathcal{F}$ implies that

$$(\mathcal{F}(\mathbf{x}))^{\mathbf{y}} = \prod_{n=1}^{N} \mathcal{F}(\mathbf{x}(n))^{\mathbf{y}(n)} = \prod_{n=1}^{N} \mathcal{F}(\mathbf{x}(n) \cdot \mathbf{y}(n)) =$$

$$= \mathcal{F}\left(\sum_{n=1}^{N} \mathbf{x}(n) \cdot \mathbf{y}(n) \right) = \mathcal{F}(\langle \mathbf{x}, \mathbf{y} \rangle) \,. \; \square$$

Protocol 3 begins with V_k submitting a query to T (Step 1); the query includes an index $n \in [N]$ of a user $u_n \in U$ and an index $m \in I_k$ of an item b_m that V_k offers. Then, T computes the set $N_q^+(m)$ (see Section 4.1); those are the indices $\{\ell_1, \ldots, \ell_t\}$ of the $t \le q$ items in $B \setminus \{b_m\}$ (not necessarily from among those offered by V_k) that have the highest and positive similarity scores against b_m (Step 2). T then sets $\mathbf{s}_m$ to be the mth column of the similarity matrix, where all entries not corresponding to the above $N_q^+(m)$-items are zeroed (Step 3). Those steps can be carried out just once for each item, so that the results can be used in future queries for the same item.

In Step 4 T selects a random multiplier g (as explained in Section 4.4) that will obfuscate from V_k the values of the numerator and the denominator in the quotient in Eq. (4), but will enable V_k to compute that quotient. Recall that, owing to Protocol 2, T has the vectors $\mathcal{F}(\hat{\mathbf{r}}_n)$ and $\mathcal{F}(\xi(\mathbf{r}_n))$ for u_n. Then, relying on Lemma 5.1, T computes in Step 5 the $\mathcal{F}$-encryption x of the scalar product $x' := g \cdot \sum_{i=1}^{t} S(\ell_i, m)\hat{\mathbf{r}}_n(\ell_i)$, as well as the $\mathcal{F}$-encryption y of the scalar product $y' := g \cdot \sum_{i=1}^{t} S(\ell_i, m)\xi(\mathbf{r}_n(\ell_i))$. It sends those values to V_k who proceeds to decrypt them (Step 6) and then use them in Eq. (4) to get the predicted rating (Step 7). (Recall that the average rating for each item b_m, $m \in I_k$, can be computed by V_k alone and was already computed in Protocol 2.)

Privacy. The only party who receives information in Protocol 3 is V_k. It receives the numerator and denominator in the quotient in Eq. (4) that determines the desired predicted rating. Since both values are obfuscated by a random multiplier that T generates, V_k only receives their quotient but it does not learn the value of neither of them beyond what is implied by that quotient.

Computational and communication costs. The computational cost for T in an execution of Protocol 3 is $2t[Exp]$ (Step 5); indeed, only t entries in the vector $g \cdot \mathbf{s}_m$ are nonzero and hence T needs to perform only t exponentiations for computing each of the scalars x and y. The computational

Protocol 3 Computing a predicted rating of $u \in U$ for an item offered by V_k.

```
1: $V_k$ sends to $T$ a query $(n, m) \in [N] \times I_k$.
2: $T$ computes the set $N_q^+(m) := \{\ell_1, \ldots, \ell_t\}$.
3: $T$ sets an $M$-dimensional vector $\mathbf{s}_m$ where $\mathbf{s}_m(\ell) = S(\ell, m)$ if $\ell \in N_q^+(m)$ and $\mathbf{s}_m(\ell) = 0$ otherwise.
4: $T$ selects a random integer multiplier $g$.
5: $T$ sends to $V_k$ the two scalar values $x = \mathcal{F}(\hat{\mathbf{r}}_n)^{(g \cdot \mathbf{s}_m)}$ and $y = \mathcal{F}(\xi(\mathbf{r}_n))^{(g \cdot \mathbf{s}_m)}$.
6: $V_k$ computes $x' := \mathcal{F}^{-1}(x)$ and $y' := \mathcal{F}^{-1}(y)$.
7: $V_k$ sets $P(u_n, b_m) = \overline{R(b_m)} + x'/Ly'$ if $y' \neq 0$, and $P(u_n, b_m) = \overline{R(b_m)}$ otherwise.
Ensure: $V_K$ gets $P(u_n, b_m)$.
```

cost for V_k is $2[Dec]$ (Step 6). Since $t \leq q$ and q is typically a small number (usually less than 20, see [55, Section 5]), the price of privacy in terms of computational overhead is very small in the online phase. As for communication, Protocol 3 consists of two rounds (Steps 1 and 5) in each of which only two scalars are transmitted.

5.4 Computing the most recommended items

Here we discuss the case where in the online phase, instead of showing to the user u_n predicted ratings for items that he had not rated so far, V_k presents to him a list of the h items, yet unrated by u_n, which are most likely to appeal to him. In view of our discussion in Section 4.1, such a computation would be carried out by V_k and T as follows: if $I(n)$ is the subset of indices of items that u_n already rated, then V_k will select the h indices $m \in I_k \setminus I(n)$ for which $\hat{s}(m) = \sum_{\ell \in I(n) \cap N_q(m)} S(\ell, m)$, Eq. (3), are largest, where $S(\ell, m)$ are the scaled integral similarity scores that T got as a result of Protocol 1.

Protocol 4 performs that computation in a privacy-preserving manner. It starts by V_k submitting a query to T (Step 1). In Steps 2-4, T generates an $\mathcal{F}$-encryption of $\hat{s}(m)$ for all $m \in I_k$, multiplied by a random integer multiplier g, as described in Section 4.4. Indeed, by Eq. (3) and Lemma 5.1, and the fact that $\xi(\mathbf{r}_n)$ is a binary vector with $\xi(\mathbf{r}_n)(\ell) = 1$ if and only if $\ell \in I(n)$, it follows that $\mathbf{x}(m) = \mathcal{F}(g\hat{s}(m))$ for all $m \in I_k$.

Next (Step 5), T computes an M_k-dimensional vector $\mathbf{y}$ such that $\mathbf{y}(m) = \mathcal{F}(1)$ if $m \in I_k \cap I(n)$ (namely, if u_n already rated that item) and $\mathbf{y}(m) = \mathcal{F}(0)$ if $m \in I_k \setminus I(n)$. T generates the vector $\mathbf{y}$ by taking the restriction of the vector $\mathcal{F}(\xi(\mathbf{r}_n))$ (that T received in Protocol 2) to I_k and multiplying each of its entries by a fresh random encryption of 0. Because $\mathcal{F}$ is homomorphic, such an operation changes only the ciphertext value but not the underlying plaintext. The reason for this seemingly redundant operation will be clarified below.

Next, T sends a random permutation of the vectors $\mathbf{x}$ and $\mathbf{y}$ to V_k, who decrypts them into $\mathbf{x}'$ and $\mathbf{y}'$, respectively (Steps 6-7). Hence, $\mathbf{x}'$ holds a permutation of the values $g\hat{s}(m)$, for all items $m \in I_k$, while $\mathbf{y}'$ holds a corresponding permutation of 0 and 1 values that identify those items which u_n had already rated. Since T had scrambled the latter ciphertexts (by multiplying each one of them with a fresh random encryption of 0), V_k can only distinguish between rated and unrated items, but it cannot distinguish between items within either one of those two subsets. (If T had not multiplied each $\mathcal{F}(\xi(\mathbf{r}_n)(m))$ with a fresh encryption of zero, then V_k would have been able to reverse engineer the random permutation π by comparing the entries of $\mathbf{y}$ with the encrypted entries of $\xi(\mathbf{r}_n)$ that it had sent to T earlier in Step 4 in Protocol 2.) V_k proceeds to find the indices of the h yet unrated items with largest $g\hat{s}(m)$ (and hence also largest $\hat{s}(m)$) and sends them to T

(Step 8). T responds by letting V_k know the original indices of those items (Step 9). Those are the items to be presented to the user u_n as the top-h recommended items.

Protocol 4 Computing for u_n the top h yet unrated items offered by V_k.

```
1: $V_k$ sends to $T$ a query $n \in [N]$.
2: For each $m \in I_k$, $T$ defines an $M$-dimensional vector $\mathbf{s}_m$ where $\mathbf{s}_m(\ell) = S(\ell, m)$ if $\ell \in N_q(m)$ and $\mathbf{s}_m(\ell) = 0$ otherwise.
3: $T$ selects a random integer multiplier $g$.
4: $T$ computes an $M_k$-dimensional vector $\mathbf{x}$ where $\mathbf{x}(m) = \mathcal{F}(\xi(\mathbf{r}_n))^{(g \cdot \mathbf{s}_m)}$, $\forall m \in I_k$.
5: $T$ computes an $M_k$-dimensional vector $\mathbf{y}$, where $\mathbf{y}(m) = \mathcal{F}(\xi(\mathbf{r}_n)(m)) \cdot \mathcal{F}(0)$, $\forall m \in I_k$.
6: $T$ generates a secret and random permutation $\pi$ over $I_k$ and sends to $V_k$ the vectors $\pi(\mathbf{x})$ and $\pi(\mathbf{y})$.
7: $V_k$ computes $\mathbf{x}' := \mathcal{F}^{-1}(\pi(\mathbf{x}))$ and $\mathbf{y}' := \mathcal{F}^{-1}(\pi(\mathbf{y}))$.
8: $V_k$ sends to $T$ the set of indices $\{m_1, \ldots, m_h\}$ in which $\mathbf{y}'(\cdot) = 0$ and $\mathbf{x}'(\cdot)$ are largest.
9: $T$ sends back to $V_k$ the set of original indices $\{\pi^{-1}(m_1), \ldots, \pi^{-1}(m_h)\}$ in a new random order.
Ensure: $V_K$ gets the indices in $I_k \setminus I(n)$ of the top $h$ items to be recommended to $u_n$.
```

Privacy. The values of the scores $\hat{s}(m)$ are hidden from V_k by the random multiplier g. However, using that mechanism alone would have leaked to V_k the ratios between the scores $\hat{s}(m)$. In order to reduce such information leakage (even though the ratio between $\hat{s}(m_1)$ and $\hat{s}(m_2)$ is arguably non-sensitive), we introduced two additional mechanisms. One is the random permutation π which prevents V_k from associating a given masked score $g\hat{s}(m)$ to an item b_m. The other mechanism is the ciphertext scrambling that T did in Step 5, as we explained above. The combination of those two mechanisms allows V_k and T to find the h items which are yet unrated by u_n and have the largest $\hat{s}(m)$ scores, without disclosing to V_k information on the scores $\hat{s}(m)$ of the items that it offers.

Computational and communication costs. The computational cost for T in an execution of Protocol 4 is $M_k q[Exp]$ (Step 4). (In Step 5 T uses encryptions of zero, $\mathcal{F}(0)$. T can compute offline a pool of such encryptions and then, whenever needed, select such values randomly from that pool.) The computational cost for V_k is $2M_k[Dec]$ (Step 7). The protocol consists of four communication rounds (Steps 1,6,8,9) in which the number of exchanged bits is $O(M_k \lambda)$, where λ is the number of bits in encrypted values.

6 PRIVACY PRESERVING ALGORITHMS FOR THE SCENARIO OF HORIZONTAL DISTRIBUTION

Here we deal with the horizontal distribution scenario. In Section 6.1 we describe the computations and protocols that are performed in the offline phase. Here, like in the vertical setting, the offline phase requires the cooperation of all parties – $V_1, \ldots, V_K$ and T, and it ends with T having the matrix of item-to-item similarity scores. Then, we describe the online phase in which a vendor V_k computes, with T's assistance, predicted ratings (Section 6.2) or the top-h recommended items (Section 6.3) in a privacy-preserving manner. The privacy goal and assumptions are as described in Section 5.

6.1 Offline model construction

That phase starts with the vendors deciding on a random ordering of both U and B; those orders are kept secret from T. Assuming that V_k serves N_k users, $k \in [K]$, where $\sum_{k \in [K]} N_k = N$, then each V_k knows the N_k rows in the user-item rating matrix R that correspond to the N_k users that it serves. Now, the similarity scores $S(\ell, m)$, $\ell \neq m \in [M]$, see Eq. (1), depend on the vectors $\mathbf{c}_m$,

$m \in [M]$, which are the columns of R. While in the vertical setting, each such column was known, in its entirety, to only one vendor, in the horizontal setting every vendor V_k knows the projection of all those vectors on its subset of N_k rows. Hence, in order to compute the similarity scores, the SMC problem which we need to solve is that of secure summation (while in the vertical setting it was the problem of SSP).

Let us denote the N_k-dimensional vectors known to V_k by $\mathbf{c}_m^{(k)}$, $m \in [M]$. Protocol 5 describes the way in which the similarity scores can be computed securely in this setting. The idea is to compute the following $M(M-1)/2$ values,

$$x(\ell, m) := \langle \mathbf{c}_\ell, \mathbf{c}_m \rangle \,, \quad 1 \le \ell < m \le M \,, \tag{5}$$

as well as the following $M(M-1)$ values,

$$y(\ell, m) := \langle \mathbf{c}_\ell^2, \xi(\mathbf{c}_m) \rangle \,, \quad 1 \le \ell \ne m \le M \,. \tag{6}$$

Given those $3M(M-1)/2$ values, we have, by Eq. (1),

$$S(\ell, m) = \frac{x(\ell, m)}{y(\ell, m)^{1/2} y(m, \ell)^{1/2}} \,, \quad 1 \le \ell < m \le M \,.$$

As each of the terms $x(\ell, m)$ and $y(\ell, m)$ can be expressed as a sum of K addends, where each addend is known to a single vendor, that computation can be carried out by invoking a secure summation protocol. To reduce the leakage of information to T, the vendors select for each pair of items, $1 \le \ell < m \le M$, an independent random integer multiplier $g_{\ell,m}$ so that T recovers the values $X(\ell, m) = g_{\ell,m} x(\ell, m)$, $1 \le \ell < m \le M$, and $Y(\ell, m) = g_{\ell,m} y(\ell, m)$, $1 \le \ell \ne m \le M$. That way, T cannot learn the values $x(\ell, m)$ and $y(\ell, m)$ but it can compute $S(\ell, m)$. As in the vertical setting, once T gets $S(\ell, m)$, $\ell, m \in [M]$, it translates it into a matrix of integers by replacing each entry σ in that matrix with $\lfloor L\sigma + 0.5 \rfloor$.

Protocol 5 Computing the matrix of item similarity scores.

```
Require: Each $V_k$, $k \in [K]$, holds a projection $\mathbf{c}_m^{(k)}$ of each of $R$'s columns, $\mathbf{c}_m$, on the subset of $N_k$
    users that it serves, $m \in [M]$.
 1: $V_1, \ldots, V_k$ select $\binom{M}{2}$ independent random integer multipliers $g_{\ell,m}$ for each pair of items
    $1 \le \ell < m \le M$.
 2: Each $V_k$ computes $X^{(k)}(\ell, m) = g_{\ell,m} \cdot \langle \mathbf{c}_\ell^{(k)}, \mathbf{c}_m^{(k)} \rangle$, $1 \le \ell < m \le M$.
 3: Each $V_k$ computes $Y^{(k)}(\ell, m) = g_{\ell,m} \cdot \langle (\mathbf{c}_\ell^{(k)})^2, \xi(\mathbf{c}_m^{(k)}) \rangle$, $1 \le \ell \ne m \le M$.
 4: $V_1, \ldots, V_K$ engage in a secure summation protocol for computing random additive shares in
    $X(\ell, m) := \sum_{k=1}^K X^{(k)}(\ell, m)$, $1 \le \ell < m \le M$, and $Y(\ell, m) := \sum_{k=1}^K Y^{(k)}(\ell, m)$, $1 \le \ell \ne m \le M$.
 5: Each $V_k$ sends its shares in those $3M(M-1)/2$ sums to $T$.
 6: $T$ adds the shares and recovers $X(\ell, m)$, $1 \le \ell < m \le M$, and $Y(\ell, m)$, $1 \le \ell \ne m \le M$.
 7: $T$ computes $S(\ell, m) = X(\ell, m)/\sqrt{Y(\ell, m) Y(m, \ell)}$ for all $1 \le \ell < m \le M$.
Ensure: $T$ gets $S(\ell, m)$.
```

We note that the offline phase in both the vertical and horizontal settings requires each pair of vendors to communicate between themselves. Interestingly, in the horizontal setting it is possible to carry out the offline phase in a manner that requires each vendor to communicate only with a single designated vendor, say V_1. Such a modification reduces the communication cost from $O(K^2)$ to $O(K)$. Such a relaxation of the communication requirements may allow larger numbers of vendors, K, to cooperate in the privacy-preserving distributed computation. To achieve that goal, we may modify Protocol 5 by replacing Steps 1 and 4-6 in it with the following ones:

- Step 1': V_1 selects $\binom{M}{2}$ independent random integer multipliers $g_{\ell,m}$ for each pair of items $1 \leq \ell < m \leq M$ and sends them to $V_2, \ldots, V_K$.
- Step 4': Each V_k sends to V_1 the values $\{\mathcal{E}(X^{(k)}(\ell,m)) : 1 \leq \ell < m \leq M\}$ and $\{\mathcal{E}(Y^{(k)}(\ell,m)) : 1 \leq \ell \neq m \leq M\}$, where $\mathcal{E}$ is a probabilistic homomorphic encryption function for which the encryption key is public but the decryption key is known only to T.
- Step 5': V_1 will then send to T the values $\{\prod_{k=1}^{K} \mathcal{E}(X^{(k)}(\ell,m)) = \mathcal{E}(X(\ell,m)) : 1 \leq \ell < m \leq M\}$ and $\{\prod_{k=1}^{K} \mathcal{E}(Y^{(k)}(\ell,m)) = \mathcal{E}(Y(\ell,m)) : 1 \leq \ell \neq m \leq M\}$.
- Step 6': T decrypts and recovers $X(\ell,m)$, $1 \leq \ell < m \leq M$, and $Y(\ell,m)$, $1 \leq \ell \neq m \leq M$.

In addition to the above mentioned advantages in terms of communication costs, such a variant of Protocol 5 may be used to hide from all vendors, apart from V_1, the number K and identity of the other vendors.

Privacy. None of the vendors learns any information during either of the two variants of Protocol 5, since they only engage in a secure summation sub-protocol (using either secret sharing in the basic Protocol 5, or $\mathcal{E}$-encryptions in the modified version, where $\mathcal{E}$ is a cipher that can be decrypted only by T). T is the only party that receives information, but owing to the usage of random multipliers, it can only deduce the desired similarity scores, but not the numerators and denominators in the quotients that define them.

Computational and communication costs. The computational complexity of Protocol 5 is low since it does not involve any of the costly operations ([*Enc*], [*Dec*], [*Exp*]). The above described modification of Protocol 5, on the other hand, imposes a computational cost of $1.5M(M-1)[Enc]$ on each vendor and a computational cost of $1.5M(M-1)[Dec]$ on T. As for the communication costs, the modified Protocol 5 consists of three rounds (Steps 1',4',5') in which the number of communicated bits is $O(KM^2\lambda)$. (Recall that, as in Section 5, λ denotes the number of bits in encrypted values.)

6.2 Online computation of predicted rating

In this phase, any vendor V_k, $k \in [K]$, can submit a query to T for the predicted rating of a user $u_n \in U$ that it serves for an item $b \in B$. Such queries are answered, in a privacy preserving manner, by Protocol 6. Here, $\mathcal{F}$ denotes a probabilistic homomorphic encryption function for which V_k knows the key pair while T knows only the encryption key. (Note that in the vertical setting it was essential that all vendors use the same encryption function $\mathcal{F}$. However, in the horizontal setting, each vendor may have its own encryption function $\mathcal{F}$. Therefore, there is no need for collaboration between the vendors in that regard.)

The protocol begins with V_k submitting a query to T (Step 1); the query includes an index $m \in [M]$ of an item $b_m \in B$ and two encrypted vectors: $\mathcal{F}(\mathbf{r}_n)$ – an encryption of u_n's row in the user-item rating matrix R, and $\mathcal{F}(\xi(\mathbf{r}_n))$ – an encryption of the boolean vector that indicates which items that particular user rated in the past. The protocol continues just like Protocol 3.

The privacy analysis and computational and communication costs analysis of Protocol 6 are similar to those of Protocol 3.

6.3 Online computation of the top h recommended items

In order to let V_k get the top-h yet unrated items for a user u_n that it serves, one can apply Protocol 4 with the following modifications: (a) In Step 1, instead of sending to T the user's index, V_k sends the two encrypted vectors of that user, $\mathcal{F}(\mathbf{r}_n)$ and $\mathcal{F}(\xi(\mathbf{r}_n))$. (b) Since in the horizontal setting V_k offers all items, then $M_k = M$ and $I_k = [M]$; hence, the vectors $\mathbf{x}$ and $\mathbf{y}$ (Steps 4-5) are M-dimensional, and the permutation π (Step 6) is over $[M]$.

Protocol 6 Computing predicted ratings of $u \in U$ for an item.

```
$V_k$ sends to $T$ an item index $m \in [M]$ and two encrypted vectors, $\mathcal{F}(\mathbf{r}_n)$ and $\mathcal{F}(\xi(\mathbf{r}_n))$.
$T$ computes the set $N_q^+(m) := \{\ell_1, \ldots, \ell_t\}$.
$T$ sets an $M$-dimensional vector $\mathbf{s}_m$ where $\mathbf{s}_m(\ell) = S(\ell, m)$ if $\ell \in N_q^+(m)$ and $\mathbf{s}_m(\ell) = 0$ otherwise.
$T$ selects a random integer multiplier $g$.
$T$ sends to $V_k$ the two scalar values $x = \mathcal{F}(\hat{\mathbf{r}}_n)^{(g \cdot \mathbf{s}_m)}$ and $y = \mathcal{F}(\xi(\mathbf{r}_n))^{(g \cdot \mathbf{s}_m)}$.
$V_k$ computes $x' := \mathcal{F}^{-1}(x)$ and $y' := \mathcal{F}^{-1}(y)$.
$V_k$ sets $P(u_n, b_m) = \overline{R(b_m)} + x'/Ly'$ if $y' \neq 0$, and $P(u_n, b_m) = \overline{R(b_m)}$ otherwise.
```
Ensure: V_K gets $P(u_n, b_m)$.

7 EXPERIMENTS

Hypotheses. Our evaluation seeks to validate the following research hypotheses:

- Applying PPCF protocols (such as ours) in distributed settings can significantly increase the accuracy of recommendations.
- Our PPCF protocols, which base their security on cryptographic means, offer significantly more accurate recommendations than those offered by state of the art protocols that achieve privacy preservation by introducing random noise.
- The runtime of our PPCF protocols is feasible for practical use.

Experimental setting. All experiments were run on a 13-inch MacBook Pro with a 3.0GHz dual-core Intel Core i7 CPU and 16GB of RAM. The algorithms were implemented in Java as an extension to the open source LibRec library[1].

Datasets. We used four publicly available datasets: MovieLens 100K, MovieLens 1M, MovieLens 20M, and FilmTrust. Table 1 reports their main characteristics: number of users N, number of items M, number of ratings *numR*, density (%) $D := \frac{numR}{NM} \times 100$, and the rating scale.

Table 1. Dataset characteristics

dataset	N	M	*numR*	density	scale
MovieLens 100K	943	1682	10^5	6.30%	[1,5]
MovieLens 1M	6040	3706	10^6	4.47%	[1,5]
MovieLens 20M	138000	27000	$2 \cdot 10^7$	0.54%	[1,5]
FilmTrust	1508	2071	35497	1.14%	[0.5,4]

Methodology. In each experiment we randomly split the *numR* input ratings into training (70%) and testing (30%) sets. We then simulated a vertical or horizontal distribution between K vendors by randomly splitting the complete user-item matrix R into K (almost) equal-sized sub-matrices as follows. If R is of dimensions $N \times M$, we split it in the vertical distribution scenario into K matrices of (almost equal) dimensions $N \times M_k$, where $M_k \in \{\lfloor M/K \rfloor, \lceil M/K \rceil\}$, $k \in [K]$, while in the horizontal case the K matrices are of dimensions $N_k \times M$, where $N_k \in \{\lfloor N/K \rfloor, \lceil N/K \rceil\}$, $k \in [K]$. We then ran the PPCF protocols on this distributed data and computed the performance of the resulting recommender system when trying to predict the ratings or rankings of the testing data from the training data. We repeated this process ten times, each time with new and independent random choices. Finally, we report the average performance over those ten runs.

[1] http://www.librec.net

Experiment 1. In this experiment we demonstrate the benefits that PPCF protocols, such as ours, bring to the collaborating vendors. Towards that end, we split the complete user-item matrix R into K sub-matrices, as described above. In each distribution scenario, each of the K vendors used only the training data in its sub-matrix in order to predict the testing data entries in its sub-matrix. Then, we computed the resulting mean absolute error (MAE) over all testing entries in all sub-matrices.

The left side of Figure 1 shows the resulting errors as obtained on each of our datasets, in the vertical distribution scenario, for $K = 1, 2, 4, 8, 16, 32$. As expected, basing predictions on training datasets of smaller sizes (as is the case when K increases) results in higher error levels. The case $K = 1$ corresponds to the centralized setting in which the entire matrix is owned by a single vendor; in that case, the resulting MAE is minimal. The case of $K = 32$ shows the highest MAE. If the $K = 32$ vendors had utilized our protocols, instead of working solitarily, then the MAE would decrease to the same level as shown for $K = 1$, since our protocols issue exactly the same results as those which would be obtained if all data was owned by a single entity.

The corresponding graphs for the horizontal distribution scenario are depicted on the right side of Figure 1, where $K = 1, 2, 4, 8, 16, 32, 64, 128, 256$. Similarly to the vertical case, basing predictions on training datasets of smaller sizes (as is the case when K increases) results in higher error levels. Interestingly however, in two of the examined datasets – FilmTrust and MovieLens 1M – the error increases relatively slowly in comparison to the vertical case, suggesting that collaboration between the vendors provides a significant merit only when the private datasets are of small sizes (namely, when K in our example is high). One possible explanation for this difference could be that it is easier to model the system by examining a dataset representing a small number of users for whom the complete data regarding the entire set of items is available, than by examining a dataset representing all users for which data is given only with respect to a subset of the itemset.

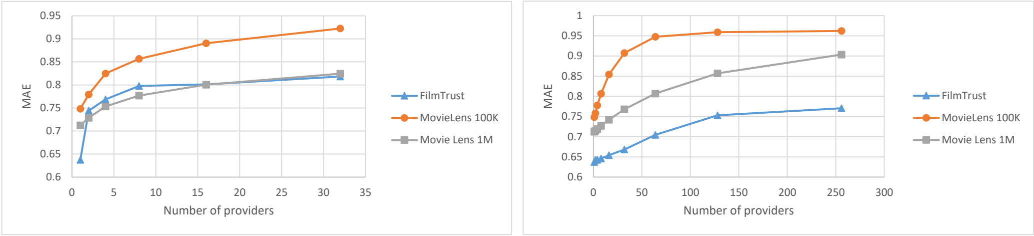

Fig. 1. Prediction errors in distributed scenarios with no collaboration — vertical (left) and horizontal (right)

In the next two experiments, we compare the performance of our protocols for predicted ratings (Experiment 2) and predicted rankings (Experiment 3) to the protocol of Yakut and Polat [55] (YP), which is the only other item-based PPCF protocol that is directly comparable to ours.

As stated earlier, our protocols for predicted ratings and rankings, in both distribution scenarios, issue the very same outputs as the basic non-private recommendation algorithm that they implement; namely, the privacy mechanisms do not alter the functionality of the recommendation algorithm. The protocol of YP, on the other hand, achieves privacy by adding noise to the data.

One of the tools which YP utilizes in order to offer privacy is by having each vendor add some fraction of fake ratings into its sub-matrix. The fake ratings replace only unrated entries in the matrix R and the relevant performance parameter in this context is the percentage p of fake ratings. Specifically, if in the vertical distribution scenario V_k has a sub-matrix of dimensions $N \times M_k$ and of the NM_k entries, only $numR_k$ are actual ratings, then V_k replaces $p\%$ of the remaining $NM_k - numR_k$ entries with fake ratings, $k = 1, 2$; similarly for the horizontal distribution scenario.

The value of the fake ratings can be set in several ways. One of the suggestions made in [55], which we adopt herein, is that each V_k uses the global mean rating over all rated entries in its sub-matrix. The manner in which fake ratings are added in the YP protocol implies that the results of the vertical split coincide with the results of the horizontal split, as is the case with our protocols.

Since the YP protocol is designed for the case of $K = 2$ only, Experiments 2 and 3 are carried out on splits of the user-item matrices into two random sub-matrices.

Experiment 2. We compared the accuracy of our Protocols 3 and 6 for predicting ratings to that of YP. We applied our protocols for each of the testing entries and then computed the MAE over all testing entries (in both sites of V_1 and V_2). We did a similar test with the prediction protocol YP, with several values of p (the percentage of fake ratings). Finally, we used a baseline algorithm that simply predicts for each user u and item b that $P(u, b) = \overline{R(b)}$, where $\overline{R(b)}$ is the average rating given to item b. Figure 2 shows the resulting errors as obtained on each of our datasets. (Recall that the shown values are averaged over ten random and independent runs.)

As can be seen, the accuracy of YP with $p = 0$ coincides with that of our Protocols 3 (in the vertical distribution scenario) and 6 (horizontal), but such a version of YP is non-private. Increasing p to higher values that would provide better hiding of each vendor's data results in higher errors, that in the two MoviLens datasets even became higher than that of the naïve (and perfectly secure) item-mean predictor. To summarize, our protocols, which base their security on cryptographic means, rather than randomization, offer higher security than YP; in addition, the output of our protocols (as opposed to YP) fully coincides with the non-secure algorithm.

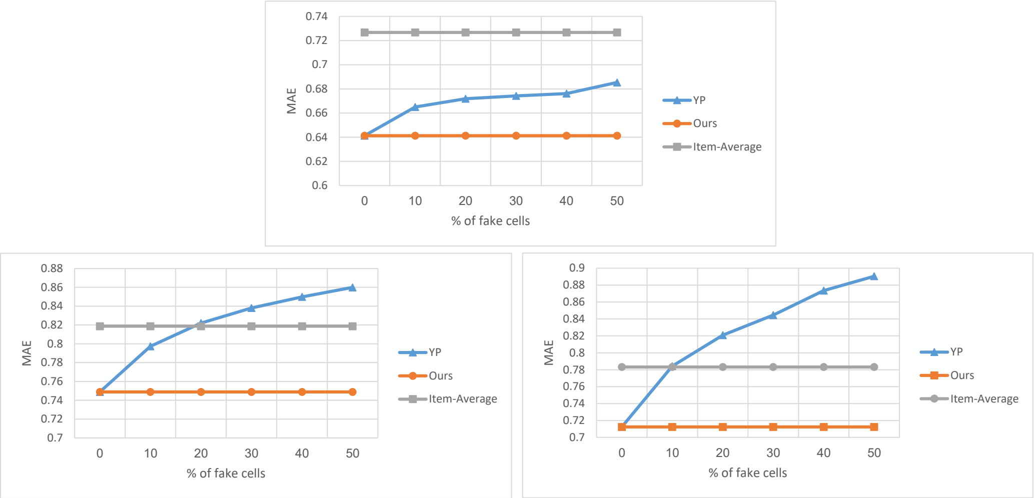

Fig. 2. Rating prediction: FilmTrust (top), MovieLens 100K (bottom left) and MovieLens 1M (bottom right)

Experiment 3. Next, we compared the quality of ranking as offered by our Protocol 4, to the quality of rankings which are derived from computing predicted ratings, by either our Protocols 3 and 6, or by YP [55]. The comparison is made by the AUC (Area Under the Receiver Operating Curve) measure; AUC values range from 0.5 (worst) to 1 (best).

Let $I_{k,n}$ denote the set of items offered by V_k which u_n had rated, and their corresponding entries in R were selected for training. We follow the usual practice of generating rankings over all items in the complement set, $I^c_{k,n} := I_k \setminus I_{k,n}$, and then comparing that ranking to the baseline ranking, in which all items in $I^c_{k,n}$ which u_n had rated are considered positive and all other items in $I^c_{k,n}$ are considered negative. Specifically, for each $m \in I^c_{k,n}$ we computed three scores: its predicted rating by Protocols 3 and 6, its predicted rating by YP, and the score $\hat{s}(m)$, Eq. (3), by which Protocol 4 ranks. By thus we get three sequences of scores which induce three rankings over $I^c_{k,n}$. We compared each of those rankings to the above described baseline ranking. Letting $E_{k,n}$ denote the AUC value obtained for one of those three rankings, we finally compute the average value $E = \left(\sum_{k=1}^{2} \sum_{n=1}^{N} E_{k,n}\right) / 2N$. Figure 3 shows the average of E, for each of the three scoring functions, over ten random runs. As can be seen, Protocol 4, which is designed for issuing rankings, issues significantly better rankings than rankings based on the predicted ratings of either our Protocols 3, or YP.

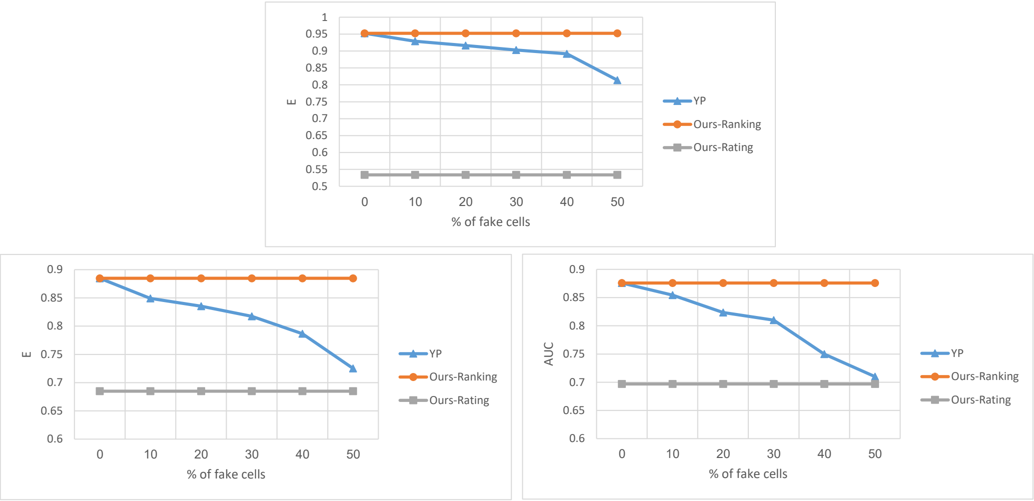

Fig. 3. Ranking prediction: FilmTrust (top), MovieLens 100K (bottom left) and MovieLens 1M (bottom right)

Experiment 4. We conclude with runtime experimentation. We measured the cryptographic overhead on both the mediator T and a typical vendor V_k in the offline and online phases. Specifically, we measured the overhead due to the privacy-induced costly operations of encryption, decryption, exponentiation (representing products in the plaintext domain) and multiplications (representing sums in the plaintext domain). For encryption we used the Paillier cryptosystem [33] implemented in Java[2]. Table 2 shows the runtime overheads in each of the four datasets, when they are distributed evenly and vertically among $K = 5$ vendors[3], in hours(h), minutes(m), seconds(s), and miliseconds(ms).

In the offline phase, we focus only on Protocol 2. We ignore Protocol 1 for two reasons. First, the actual runtime depends on the selection of the SSP sub-protocol. Second, if one chooses the SSP protocol of Du and Zhan [11], which is most fitting in our setting, then it involves no cryptographic operations, and hence the privacy-induced overhead is negligible. (Namely, the runtime of Protocol 1 is on par with that of a non-secure protocol for computing the similarity scores.) Protocol 2 entails cryptographic overhead only on the vendors, as they need to encrypt their matrix entries.

[2] www.csee.umbc.edu/~kunliu1/research/Paillier.html

[3] The times in the Rating column are independent of K. Those in the Offline and Ranking columns are inversely proportional to K, under the assumption of even distribution.

The overhead is considerable. In the three datasets apart from the largest one, MovieLens 20M, the offline computation time takes from 14.5 minutes up to 3.4 hours. However, as that phase is run only once in a period (say, once a week), and since the required computations can be easily parallelized, such overhead is reasonable.

In the largest dataset, MovieLens 20M, the situation is different. A direct application of Protocol 2 would take several days. Hence, for datasets of such dimensions, we perform the following economization. The values that V_k has to encrypt in Protocol 2 are $\hat{R}(n,m)$ (see its definition in Step 3 of Protocol 2) and $\xi(R(n,m))$, for all $n \in [N], m \in I_k$. But 99.46% of those values are just zero, as implied by the sparsity of the MovieLens 20M dataset (see Table 1). While we stressed earlier the importance of using a probabilistic encryption function that would hide from T the pattern of plaintexts underneath the encryption, performing such a large number of repeated encryptions is too extreme. (A direct application of Protocol 2 would encrypt the value zero $1.48 \cdot 10^9$ times, since that is the number of zeros in the matrices $\hat{R}(n,m)$ and $\xi(R(n,m))$ for each vendor in the vertical distribution scenario that we consider for that dataset.)

In order to dramatically decrease the runtime, we created, for each vendor, a "dictionary" of $\sqrt{2NM_k}$ random and independent encryptions of zero. Then, whenever we had to create a random encryption of zero, we selected two elements from the dictionary and multiplied them (where the selection was made uniformly at random). That way, we were able to generate the needed number of zero encryptions, with negligible probability of obtaining the same ciphertext twice. Since the cost of multiplication is much smaller than that of encryption, the resulting runtime was reduced to only 48 minutes. (This runtime is marked in the table by (*), in order to indicate that it was achieved with the above described economization.)

For the online phase, we used $q = 80$. (Recall that q is the size of the item-neighborhoods which are used in predicting ratings and rankings.) The overhead for predicting ratings (Protocol 3) is negligible. The overhead for ranking (Protocol 4) is much larger, since for ranking, it is needed to compute a score for all items (while in rating a score needs to be computed only for a single item). But despite the fact that the runtimes for computing rankings are high (few seconds in the three smaller datasets and over one minute, in total, in the largest dataset), such runtimes are no show-stoppers for three reasons: (a) The computations of T as well as those of V_k can be parallelized so that by dedicating several machines for performing the costly cryptographic operations it is possible to reduce the computation time. (b) Presenting the top h recommended items for a user may be a service which is offered at the initiation of the vendor (as opposed to one that is triggered by a demand of the user). In such a case, once the vendor identifies a user as an active consumer who should be presented with such recommendations, it may start this computation and present the issued recommendations to the user when they become available. (c) V_k may define upfront for u_n a subset of items $I_k(n) \subset I_k$ that could be of interest for him (say, based on his history or demographics) and then notify T of that subset. Then, T could compute $\mathbf{x}(m)$ and $\mathbf{y}(m)$ only for $m \in I_k(n)$. The resulting runtime, for both T and V_k, will consequently reduce by a factor of $M_k/|I_k(n)|$.

Table 2. Cryptographic runtime overheads in the vertical setting

Dataset	Offline	Rating		Ranking	
	V_k	T	V_k	T	V_k
MovieLens 100K	14.5m	13.1ms	5.1ms	2.6s	1.7s
MovieLens 1M	3.4h	15.8ms	5.1ms	5.8s	3.8s
MovieLens 20M	48m(*)	15.1ms	5.1ms	42.8s	34s
FilmTrust	37.2m	13.4ms	5.1ms	3.3s	2.1s

In the horizontal scenario, the offline phase is much faster as it involves no costly operations (see the discussion of Protocol 5); we therefore ignore it. Table 3 shows the runtime overheads in the online phase for each of the four datasets, when they are distributed evenly and horizontally among $K = 5$ vendors. The runtime for computing predicted ratings is the same as in the vertical setting. On the other hand, when computing predicted ranking, the runtimes in the horizontal scenario differ from those in the vertical scenario by a factor of $K = 5$. The manners which we discussed above for dealing with the non-negligible runtimes for producing predicted rankings are relevant in this setting as well; hence, computing predicted ranking is feasible also in this setting.

Table 3. Cryptographic runtime overheads in the horizontal setting

Dataset	Rating		Ranking	
	T	V_k	T	V_k
MovieLens 100K	13.1ms	5.1ms	13s	8.5s
MovieLens 1M	15.8ms	5.1ms	29s	19s
MovieLens 20M	15.1ms	5.1ms	214s	170s
FilmTrust	13.4ms	5.1ms	16.5s	10.5s

8 CONCLUSION

We devised herein secure multi-party protocols for executing item-based PPCF over distributed datasets, for both the vertical and horizontal distributed settings. Our protocols rely on a mediator. Such a mediator is essential since, without it, the different vendors would need to constantly be online and be ready to serve requests by other vendors. Our protocols issue exactly the same results as their non-privacy preserving counterparts, and they protect each vendor's data from other vendors as well as from the mediator. Our protocols rely solely on existing cryptographic arsenal; this offers a significant advantage as they can be readily implemented on top of standard libraries.

We note that the offline phase, in either of the distribution scenarios, is applied on the current snapshot of the user-item matrix R. It may be repeated periodically (say, once a week) in order to update the similarity scores according to the changes in the user set U, item set B or the user-item matrix R. Even though computations from a previous offline phase can be used for the next offline phase (for example, all entries in R that were not changed do not need to be encrypted again), we suggest to run all computations in the offline phase from scratch. By doing so, the mediator cannot compare its view in one offline phase to its view in a previous offline phase in order to extract inferences on the differences that occurred in R, since the encryption functions are probabilistic. The runtimes which we reported in Section 7 show that such a practice is clearly feasible and there is no point in exercising economizations that may jeopardize privacy.

This study suggests several future research directions:

(a) Generalizing our protocols to deal with an arbitrary distribution setting (i.e, not a purely vertical or a purely horizontal split).

(b) Examining the applicability of our techniques to other CF algorithms, such as matrix factorization-based algorithms and their extensions (e.g. the one in [39]) and compare their performance to that of existing privacy-preserving matrix factorization algorithms, such as the ones in [32] and [31].

(c) Enhancing our protocols so that they offer a high level of privacy even if some of the interacting parties do not act honestly, or collude. As explained in Section 5, there are substantial reasons to accept the assumption that all parties involved are semi-honest. However, a corrupted mediator T may bribe one of the vendors and thus obtain the decryption key of the encryption function $\mathcal{F}$; by doing so, T may reveal private information of all vendors. We deem this option highly improbable, since if T cheats it risks losing the trust vested in it, and then losing the reputation on which its business or public status depend. But it is still worthwhile to add such defense mechanisms, if only to increase the trust of the vendors in the security of the system. Hence, in future work we intend to split the role of T into k distinct and independent mediators, $T_1, \ldots, T_k$, and enhance our protocols by secret sharing techniques, so that the recovery of the final rating or ranking results requires a collaboration of all k mediators, or of k' mediators, where $1 < k' < k$ (the latter option increases the robustness of the system). These settings may significantly reduce the chances of corruption since it would be then necessary for k' mediators to collaborate in attempting to gain access to private information of the vendors. To the best of our knowledge, none of the existing PPCF studies had addressed such concerns and offered corresponding remedies.

REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. 2000. Privacy Preserving Data Mining. In *SIGMOD Conference*. 439–450.

[2] Joël Alwen, Jonathan Katz, Yehuda Lindell, Giuseppe Persiano, Abhi Shelat, and Ivan Visconti. 2009. Collusion-Free Multiparty Computation in the Mediated Model. In *CRYPTO*. 524–540.

[3] Joël Alwen, Abhi Shelat, and Ivan Visconti. 2008. Collusion Free Protocols in the Mediated Model. In *CRYPTO*. 497–514.

[4] Abdelrahaman Aly, Edouard Cuvelier, Sophie Mawet, Olivier Pereira, and Mathieu Van Vyve. 2013. Securely Solving Simple Combinatorial Graph Problems. In *FC*. 239–257.

[5] Gilad Asharov, Francesco Bonchi, David García-Soriano, and Tamir Tassa. 2017. Secure Centrality Computation Over Multiple Networks. In *WWW*. 957–966.

[6] Anirban Basu, Jaideep Vaidya, and Hiroaki Kikuchi. 2012. Perturbation Based Privacy Preserving Slope One Predictors for Collaborative Filtering. In *IFIPTM*. 17–35.

[7] Justin Brickell and Vitaly Shmatikov. 2005. Privac Preserving Graph Algorithms in the Semi-honest Model. In *ASIACRYPT*. 236–252.

[8] Iván Cantador, Ignacio Fernández-Tobías, Shlomo Berkovsky, and Paolo Cremonesi. 2015. Cross-Domain Recommender Systems. In *Recommender Systems Handbook*. 919–959.

[9] Fran Casino, Constantinos Patsakis, Domenec Puig, and Agusti Solanas. 2013. On Privacy Preserving Collaborative Filtering: Current Trends, Open Problems, and New Issues. In *ICEBE*. 244–249.

[10] Keke Chen and Ling Liu. 2005. Privacy Preserving Data Classification with Rotation Perturbation. In *ICDM*. 589–592.

[11] Wenliang Du and Justin Zhijun Zhan. 2002. A Practical Approach to Solve Secure Multi-party Computation Problems. In *Workshop on New Security Paradigms*. 127–135.

[12] Cynthia Dwork. 2006. Differential Privacy. In *ICALP (2)*. 1–12.

[13] Michael D. Ekstrand, John Riedl, and Joseph A. Konstan. 2011. Collaborative Filtering Recommender Systems. *Foundations and Trends in Human-Computer Interaction* 4, 2 (2011), 175–243.

[14] Zekeriya Erkin, Thijs Veugen, Tomas Toft, and Reginald L. Lagendijk. 2013. Privacy Preserving Distributed Clustering. *EURASIP J. Information Security* 2013 (2013), 4.

[15] Arik Friedman, Bart Knijnenburg, Kris Vanhecke, Luc Martens, and Shlomo Berkovsky. 2015. Privacy Aspects of Recommender Systems. In *Recommender Systems Handbook*. 649–688.

[16] Christos K. Georgiadis, Nikolaos Polatidis, Haralambos Mouratidis, and Elias Pimenidis. 2017. A Method for Privacy Preserving Collaborative Filtering Recommendations. *The Journal of Universal Computer Science* 23 (2017), 146–166.

[17] David Goldberg, David A. Nichols, Brian M. Oki, and Douglas B. Terry. 1992. Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM* 35, 12 (1992), 61–70.

[18] Tal Grinshpoun and Tamir Tassa. 2014. A Privacy Preserving Algorithm for Distributed Constraint Optimization. In *AAMAS*. 909–916.

[19] Tal Grinshpoun and Tamir Tassa. 2016. P-SyncBB: A Privacy Preserving Branch and Bound DCOP Algorithm. *Journal of Artificial Intelligence Research* 57 (2016), 621–660.

[20] Tal Grinshpoun, Tamir Tassa, Vadim Levit, and Roie Zivan. 2019. Privacy Preserving Region Optimal Algorithms for Symmetric and Asymmetric DCOPs. *Artificial Intelligence* 266 (2019), 27–50.

[21] Arjan Jeckmans, Qiang Tang, and Pieter H. Hartel. 2012. Privacy Preserving Collaborative Filtering Based on Horizontally Partitioned Dataset. In *International Conference on Collaboration Technologies and Systems*. 439–446.

[22] Somesh Jha, Louis Kruger, and Patrick D. McDaniel. 2005. Privacy Preserving Clustering. In *ESORICS*. 397–417.

[23] Wei Jiang and Chris Clifton. 2006. A Secure Distributed Framework for Achieving k-Anonymity. *The VLDB Journal* 15 (2006), 316–333.

[24] Murat Kantarcioglu and Chris Clifton. 2004. Privacy Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. *IEEE Transactions on Knowledge and Data Engineering* 16 (2004), 1026–1037.

[25] Thomas Léauté and Boi Faltings. 2013. Protecting Privacy through Distributed Computation in Multi-agent Decision Making. *Journal of Artificial Intelligence Research* 47 (2013), 649–695.

[26] Daniel Lemire and Anna Maclachlan. 2005. Slope One Predictors for Online Rating-Based Collaborative Filtering. In *SDM*. 471–475.

[27] Xiaodong Lin, Chris Clifton, and Michael Y. Zhu. 2005. Privacy Preserving Clustering with Distributed EM Mixture Modeling. *Knowledge and Information Systems* 8 (2005), 68–81.

[28] Yehuda Lindell and Benny Pinkas. 2000. Privacy Preserving Data Mining. In *Crypto*. 36–54.

[29] Olvi L. Mangasarian and Edward W. Wild. 2008. Privacy Preserving Classification of Horizontally Partitioned Data via Random Kernels. In *DMIN*. 473–479.

[30] Olvi L. Mangasarian, Edward W. Wild, and Glenn Fung. 2008. Privacy Preserving Classification of Vertically Partitioned Data via Random Kernels. *ACM Transactions on Knowledge Discovery from Data* 2, 3 (2008), 12:1–12:16.

[31] Kartik Nayak, Xiao Shaun Wang, Stratis Ioannidis, Udi Weinsberg, Nina Taft, and Elaine Shi. 2015. GraphSC: Parallel Secure Computation Made Easy. In *IEEE Symposium on Security and Privacy*. 377–394.

[32] Valeria Nikolaenko, Stratis Ioannidis, Udi Weinsberg, Marc Joye, Nina Taft, and Dan Boneh. 2013. Privacy Preserving Matrix Factorization. In *CCS*. 801–812.

[33] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Eurocrypt*. 223–238.

[34] Huseyin Polat and Wenliang Du. 2003. Privacy Preserving Collaborative Filtering Using Randomized Perturbation Techniques. In *ICDM*. 625–628.

[35] Huseyin Polat and Wenliang Du. 2005. Privacy Preserving Collaborative Filtering. *Int. J. Electronic Commerce* 9 (2005), 9–35.

[36] Huseyin Polat and Wenliang Du. 2005. Privacy Preserving Collaborative Filtering on Vertically Partitioned Data. In *PKDD*. 651–658.

[37] Huseyin Polat and Wenliang Du. 2008. Privacy Preserving Top-N Recommendation on Distributed Data. *Journal of the Association for Information Science and Technology* 59 (2008), 1093–1108.

[38] Nikolaos Polatidis, Christos K. Georgiadis, Elias Pimenidis, and Haralambos Mouratidis. 2017. Privacy Preserving Collaborative Recommendations Based on Random Perturbations. *Expert Systems with Applications* 71 (2017), 18–25.

[39] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[40] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item Based Collaborative Filtering Recommendation Algorithms. In *WWW*. 285–295.

[41] J. Ben Schafer, Joseph A. Konstan, and John Riedl. 1999. Recommender Systems in E-Commerce. In *EC*. 158–166.

[42] Assaf Schuster, Ran Wolff, and Bobi Gilburd. 2004. Privacy Preserving Association Rule Mining in Large Scale Distributed Systems. In *CCGRID*. 411–418.

[43] Erez Shmueli and Tamir Tassa. 2017. Secure Multi-Party Protocols for Item-Based Collaborative Filtering. In *RecSys*. 89–97.

[44] Tamir Tassa. 2014. Secure Mining of Association Rules in Horizontally Distributed Databases. *IEEE Transactions on Knowledge and Data Engineering* 26 (2014), 970–983.

[45] Tamir Tassa and Francesco Bonchi. 2014. Privacy Preserving Estimation of Social Influence. In *EDBT*. 559–570.

[46] Tamir Tassa and Dror J. Cohen. 2013. Anonymization of Centralized and Distributed Social Networks by Sequential Clustering. *IEEE Transactions on Knowledge and Data Engineering* 25 (2013), 311–324.

[47] Tamir Tassa, Tal Grinshpoun, and Avishay Yanai. 2019. A Privacy Preserving Collusion Secure DCOP Algorithm. In *IJCAI*.

[48] Tamir Tassa, Tal Grinshpoun, and Roie Zivan. 2017. Privacy Preserving Implementation of the Max-Sum Algorithm and its Variants. *Journal of Artificial Intelligence Research* 59 (2017), 311–349.

[49] Tamir Tassa and Ehud Gudes. 2012. Secure Distributed Computation of Anonymized Views of Shared Databases. *Transactions on Database Systems* 37, Article 11 (2012).

[50] Tamir Tassa, Ayman Jarrous, and Yonatan Ben-Ya'akov. 2013. Oblivious Evaluation of Multivariate Polynomials. *Journal of Mathematical Cryptology* 7 (2013), 1–29.

[51] Tamir Tassa, Roie Zivan, and Tal Grinshpoun. 2015. Max-Sum Goes Private. In *IJCAI*. 425–431.

[52] Tamir Tassa, Roie Zivan, and Tal Grinshpoun. 2016. Preserving Privacy in Region Optimal DCOP Algorithms. In *IJCAI*. 496–502.

[53] Jaideep Vaidya and Chris Clifton. 2002. Privacy preserving association rule mining in vertically partitioned data. In *KDD*. 639–644.

[54] I-Cheng Wang, Chih-Hao Shen, Justin Zhan, Tsan-sheng Hsu, Churn-Jung Liau, and Da-Wei Wang. 2009. Toward Empirical Aspects of Secure Scalar Product. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 39 (2009), 440–447.

[55] Ibrahim Yakut and Huseyin Polat. 2012. Arbitrarily Distributed Data Based Recommendations with Privacy. *Data & Knowledge Engineering* 72 (2012), 239–256.

[56] Andrew C. Yao. 1982. Protocols for Secure Computation. In *FOCS*. 160–164.

[57] Justin Zhijun Zhan, Stan Matwin, and LiWu Chang. 2005. Privacy Preserving Collaborative Association Rule Mining. In *Data and Applications Security*. 153–165.

[58] Sheng Zhong, Zhiqiang Yang, and Rebecca N. Wright. 2005. Privacy Enhancing k-Anonymization of Customer Data. In *PODS*. 139–147.

[59] Tianqing Zhu, Yongli Ren, Wanlei Zhou, Jia Rong, and Ping Xiong. 2014. An Effective Privacy Preserving Algorithm for Neighborhood Based Collaborative Filtering. *Future Generation Computer Systems* 36 (2014), 142–155.

Received XXX 2018