Privacy Preserving Collaborative Filtering by Distributed Mediation

TAMIR TASSA*, The Open University, Israel
ALON BEN HORIN, The Open University, Israel

Recommender systems have become very influential in our everyday decision making, e.g., helping us choose a movie from a content platform, or offering us suitable products on e-commerce websites. While most vendors who utilize recommender systems rely exclusively on training data consisting of past transactions that took place through them, it would be beneficial to base recommendations on the rating data of more than one vendor. However, enlarging the training data by means of sharing information between different vendors may jeopardize the privacy of users. We devise here secure multi-party protocols that enable the practice of Collaborative Filtering (CF) in a manner that preserves the privacy of the vendors and users. Shmueli and Tassa [38] introduced privacy-preserving protocols of CF that involved a mediator; namely, an external entity that assists in performing the computations. They demonstrated the significant advantages of mediation in that context. We take here the mediation approach into the next level by using several independent mediators. Such distributed mediation maintains all of the advantages that were identified by Shmueli and Tassa, and offers additional ones, in comparison with the single-mediator protocols: stronger security and dramatically shorter runtimes. In addition, while all prior art assumed limited and unrealistic settings, in which each user can purchase any given item through only one vendor, we consider here a general and more realistic setting, which encompasses all previously considered settings, where users can choose between different competing vendors. We demonstrate the appealing performance of our protocols through extensive experimentation.

CCS Concepts: • **Information systems** → **Collaborative filtering**; • **Security and privacy** → **Privacy-preserving protocols**.

Additional Key Words and Phrases: Recommender systems, Collaborative filtering, Distributed computing, Privacy, The mediated model

ACM Reference Format:
Tamir Tassa and Alon Ben Horin. 2022. Privacy Preserving Collaborative Filtering by Distributed Mediation. *ACM Trans. Intell. Syst. Technol.* 1, 1 (May 2022), 26 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Collaborative Filtering (CF) is one of the main methods used by recommender systems in order to assist users to navigate through the dazzling abundance of items (products, services, information) available to them, and find the most suitable ones for their tastes and needs [13]. In that method, predictions about the interests of a user are based on aggregated information on preferences of a large set of users. One of the most widely used approaches in CF is to base predictions on characteristics of items. In this so-called *item-based* approach, one uses the collaborative data in

*Corresponding author

Authors' addresses: Tamir Tassa, The Open University, Ra'anana, Israel, tamirta@openu.ac.il; Alon Ben Horin, The Open University, Ra'anana, Israel, alonibh@gmail.com.

2157-6904/2022/5-ART $15.00
https://doi.org/0000001.0000001

order to learn a model of similarity between items; consequently, users are offered items that are similar to previous items that they had already purchased and liked.

To improve the quality of predictions, larger volumes of training data are needed. Hence, it is in the interest of vendors to collaborate and conjoin their historical data in order to issue more accurate recommendations [7]. However, such collaboration may jeopardize the privacy of users, who trust the vendors through whom they purchased or rated items to keep that information confidential. Additionally, the vendors themselves may wish to keep their historical data for themselves, as it has commercial value that they would not like to share with potential competitors. Privacy-Preserving Collaborative Filtering (PPCF) addresses those issues by enabling the use of CF without disclosing private information.

In this study we propose secure protocols of Multi-Party Computation (MPC) [51] for item-based CF. Our protocols enable the computation of the similarities between items, and then to issue predictions of two types: how a given user would rate a given item, and what are the currently top items to be recommended to a given user.

Shmueli and Tassa [37, 38] have presented such protocols in the mediated model. In that model [2], there exists a mediator that assists in performing intermediate computations, but he is prevented from accessing the actual data due to privacy concerns. Their protocols rely on a *single* mediator, to whom the vendors provide the user-item ratings under homomorphic encryption. Owing to the homomorphic property, the mediator is capable of performing computations on the encrypted data, but thanks to the encryption he remains oblivious to the plaintext underneath. As explained in [38, Section 3], mediation is most advantageous for PPCF: it frees the vendors from the need to communicate with each other, and the need to be constantly online in order to assist other vendors in their recommendation queries; it reduces communication and computational costs; and it enables an economically-realistic collaboration model between vendors that differ in their contribution to the CF training data (as each vendor offers a different set of items to a different set of users), and in their demand from the CF system (in terms of the number and type of queries that they submit).

In this study we also present PPCF protocols in the mediated model. We implement the very same item-based CF technique as in [38]; i.e., we compute the same similarity model, and issue predicted ratings and ranking based on the same equations. However, while the protocols in [38] used a single mediator, ours rely on several independent mediators. The advantages of distributed mediation are very significant, as we proceed to explain.

First, while the single-mediator protocols relied on homomorphic encryption, the distributed mediation-based protocols that we present here rely on secret sharing as the cryptographic protection shield. Namely, each vendor creates secret shares in his own user-item rating data and distributes them among the mediators. As secret sharing is a linear operation, while homomorphic encryption requires expensive modular exponentiations, the effects on runtime costs are overwhelming. Another aspect that contributes to reducing runtime and communication costs is the size of the underlying arithmetic. Secret sharing can be executed on standard arithmetic (say, 32- or 64-bit) because it can be executed over any field that is large enough to represent all possible secret values. However, homomorphic encryption requires arithmetic of at least 512 and preferably 1024 bits.

Second, the protocols in [38] are vulnerable to malicious collusion, in the sense that if the single mediator colludes with one of the vendors, all private information is revealed to them. In the secret sharing-based protocols that we present here, a similar privacy collapse occurs only if at least half of the mediators betray the trust vested in them and collude. In case the group of mediators has an honest majority, the private information remains fully protected. In addition, the protection offered by secret sharing schemes, as opposed to that offered by homomorphic encryption, is in the highest information-theoretic sense. It means that even if the mediators would have an unlimited

computational power, they would not be able to infer any information on the private data, as long as they have an honest majority.

Another advantage that our protocols offer is that they free the vendors from any need to communicate with each other. While non-mediated PPCF protocols require a constant communication between the collaborating vendors, and the single-mediator protocols [38] reduced the communication demands only to the offline (and less frequent) phase, our protocols free the vendors from any need of communicating with each other, or even being aware of the number or the identity of other vendors. They only need to communicate with the mediators.

The last contribution that we offer is with regard to the collaborative setting, in the sense of how the user-item rating data is distributed among the vendors. All existing works on PPCF assumed distribution scenarios in which each entry in the user-item rating matrix is owned by just one vendor. Such exclusivity can be found only in unrealistic markets of zero competition: a user who wants to purchase a specific item can do so only through a single vendor. In reality, however, users usually have a choice between competing vendors. We introduce here a much more realistic distribution scenario, which generalizes all previously-considered scenarios, that allows such competition.

The outline of the paper is as follows. We begin with an overview of related work in Section 2, and preliminary discussions and necessary background in Section 3. We present our protocols in Section 4, and demonstrate their performance in Section 5. We conclude in Section 6.

2 RELATED WORK

The literature of PPCF may be classified according to their approach towards achieving privacy.

Obfuscation-based methods. In obfuscation-based methods, the user-item data is transformed in a manner that prevents individual users from being identified, while maintaining a useful level of accuracy in the generated recommendations.

Polat and Du [32] provided a solution for PPCF by disguising the users' personal data using randomized perturbation techniques. The perturbed data is sent to a data collector for further processing. The perturbations prevent the data collector from deriving explicit information about specific users, but he can still use the perturbed data in order to perform CF. Polat and Du claim that even though each user's information is scrambled, then given a sufficient number of users they are still able to produce decent rating estimations. Their rating predictions are based on the same cosine similarity score as we use in the present work.

Similarly, Yakut and Polat [50] proposed a privacy-preserving implementation of item-based CF which is also based on the cosine similarity score. They considered the case of two vendors, with a hybrid distribution scenario, and achieved privacy by injecting fake ratings into the distributed matrix of user-item ratings.

In contrast to the two previously discussed studies, in which randomness is added to the user-item data, Kikuchi and Mochizuk [21] applied randomness to the system response in order to prevent the preferences of individual users from being inferred from the system's responses, while allowing the users to calculate the exact response using the randomized one.

Another example is the study of Weinsberg et al. [49]: in their study, the goal was to prevent recommender systems from inferring the gender of users, while having an insignificant effect on the accuracy of recommendations issued to them. They first evaluated several gender inference algorithms and showed their ability to estimate with decent accuracy the gender of the user. Then, they presented techniques for slightly altering the ratings of the users, while barely affecting the recommendations given to them.

Obfuscation-based methods are scalable since the transformations usually only need to be applied to the data at the point of origin, after which the obfuscated data can be used directly. However, the security of these techniques is harder to prove since they rely on randomness and anonymity. Also, they issue inaccurate recommendations as those are computed from obfuscated data.

Clustering-based methods. Clustering-based methods rely on grouping the users into clusters and then extracting a representation of that cluster to be used in the CF process. Such methods provide anonymity for the users within each cluster. Another advantage of this approach is scalability, since a reduced representation of the data is used. Larger clusters enable smaller representations, but, on the other hand, they provide less accurate filtering. Examples of studies that suggest clustering-based PPCF methods are [9, 20, 26, 40].

Memis and Yakut [26] studied PPCF over partitioned data with overlaps between two parties. They examined PPCF solutions for both user-based and item-based CF algorithms. Specifically, the user-based CF algorithm is based on Pearson similarities (which is closely related to the cosine similarity score), while the item-based one is based on Slope-one predicator. In the user-based scenario, they initially inserted in random locations votes that could be the row mean, column mean, or the overall mean from the available ratings of the relevant vendor. Then, in order to privately compute the similarity scores between pairs of users, they used Paillier's homomorphic encryption. The procedure in the item-based scenario is quite similar, only that the similarity scores are computed between items, and not between users, and the underlying score is determined by the Slope-One predictor.

The work of Chow et al. [9] presented a way to implement CF by clustering the users and inserting fake ratings. In the clustering phase, each user uses a public hashing algorithm, computes the hash value for his movie ratings, and uploads the hash value to the server. The server partitions the set of users into buckets of similar users, where two users are in the same bucket if their hash value matches. In the recommendation stage, each user adds noise to his vector of ratings by adding artificially rated items, and he then uploads the augmented vector to the server. Consequently, the predicted rating for a given movie will be its average rating over the set of users who are similar to the user. The insertion of fake ratings naturally reduces the accuracy of this method.

The work of Shokri et al. [39] suggests achieving privacy in PPCF by means which resemble clustering. In their proposal, each user merges his profile with the profiles of similar users, through direct contact with them, before uploading it to the server that performs the CF computations.

Cryptography-based methods. In that approach, cryptographic means are used in order to protect the sensitive data. Typically, such methods use homomorphic encryption, since such encryption allows performing arithmetic computations on the encrypted values. The study of [38] that we discussed in the Introduction falls under this category.

Another cryptographic-based PPCF method was presented by Basu et al. [4]. They proposed a privacy-preserving algorithm that is based on the Slope One predictor [24]. Their way of preserving privacy includes adding random noise to the ratings, and they show that the Slope One predictor is very well suited for their purposes as it is robust to certain types of noise. They have also stated that in order to increase the level of privacy, they can combine the latter approach with homomorphic public-key encryption, so that each prediction query and response will be encrypted.

The studies of [5, 6] present a practical implementation of a PPCF system, based on the Google App Engine for Java (GAE/J) cloud platform. They designed algorithms that rely on a homomorphic encryption scheme to preserve the privacy of user data in the cloud. Their algorithm consists of two stages – an offline pre-computation stage, and online prediction stage. In the offline stage, the plaintext deviation matrix of item ratings (indicating the similarity between pairs of items) and the plaintext cardinality matrix (the number of users who have rated both items) are computed. In the

prediction stage, the user queries the cloud with his encrypted and complete rating vector. Using that vector, the Google App Engine computes encrypted predictions of ratings, based on the Slope One scheme, which only the user can decrypt.

Ahmad et al. [1] introduced the notion of *distributed trust*, which is similar to the distributed mediation model of computation that underlies our protocols. In contrast to our work, which enables multiple vendors to collaborate and produce better predictions through distributed mediation, they aimed at increasing the trust given by the users in a specific vendor. They do so by allocating several independent servers to that vendor and then having the vendor's users distribute their data between those servers, instead of submitting their private data to a single server. They provided a solution to the CF problem by the use of crossing minimization-based biclustering, and achieved privacy by using a threshold homomorphic cryptosystem with distributed key generation.

As we show here, while trust/security is the primary motivation for distributing mediation, such distribution ushers in an even more meaningful advantage: significantly reduced runtimes. That advantage is critical for the practicality of cryptography-based PPCF methods. While such methods provide strong security guarantees, without sacrificing the accuracy of their results, they do not scale well. As practical systems involve millions of users and items, such techniques may have impractical runtimes, especially in online settings where a short response time is needed. Distributed mediation enables to replace expensive homomorphic encryption with much more efficient cryptographic techniques (secret sharing), and thus it enables significantly shorter runtimes that could be a key factor in making PPCF a viable practice.

3 PRELIMINARIES

We begin this section by providing the necessary background on item-based CF (Section 3.1). We proceed to describe distributed scenarios, one of which is the general distribution scenario that we consider here (Section 3.2). Next, we describe the setting of distributed mediation (Section 3.3), and conclude with an overview of secret sharing (Section 3.4).

3.1 Item-based collaborative filtering

We provide here a brief introduction to item-based CF [11]. In what follows, we use the following notation agreements:

(1) If r is a non-negative integer then $\xi(r) = 0$ if $r = 0$ and $\xi(r) = 1$ otherwise.
(2) If $\mathbf{x}$ is a vector and f is any scalar function, then $f(\mathbf{x})$ is the vector in which $f(\mathbf{x})(\cdot) = f(\mathbf{x}(\cdot))$.
(3) If $\mathbf{x}$ and $\mathbf{y}$ are two N-dimensional vectors then $\mathbf{x} \cdot \mathbf{y}$ is the N-dimensional vector in which $(\mathbf{x} \cdot \mathbf{y})(n) = \mathbf{x}(n) \cdot \mathbf{y}(n)$, $n \in [N] := \{1, \ldots, N\}$.
(4) Inner products will be denoted by $\langle \cdot, \cdot \rangle$; the induced norm will be denoted by $\| \cdot \|$.

Let $U = \{u_1, \ldots, u_N\}$ be a set of users (consumers) and $B = \{b_1, \ldots, b_M\}$ be a set of items (products or services). The user-item rating matrix, R, is an $N \times M$ matrix where $R(n, m)$ is either a positive integer which indicates a rating that u_n had given to b_m, or zero if u_n had not rated b_m. In item-based CF, one uses the rating information, as given in R, in order to infer similarities between the items. This similarity model is then used in order to predict how users would rate items that they still had not purchased, or to determine the potentially most appealing items for a given user.

Let S be a symmetric $M \times M$ matrix where $S(\ell, m)$ is the similarity score between items b_ℓ and b_m, $\ell, m \in [M] := \{1, 2, \ldots, M\}$. Then the similarity scores are defined in Definition 3.1.

Definition 3.1. *Let $\mathbf{c}_m = (R(n, m) : n \in [N])$ denote the m-th column in the user-item rating matrix R, where $m \in [M]$. Given indices of two items, $\ell, m \in [M]$, let $\mathbf{c}_{\ell|m} := \mathbf{c}_\ell \cdot \xi(\mathbf{c}_m)$ denote the projection of the ℓ-th 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 n-th row of the user-item rating matrix R.
- $\overline{\mathbf{r}}_n$ be the vector of u_n's adjusted ratings, i.e. $\overline{\mathbf{r}}_n(\ell) = (R(n, \ell) - \overline{R(b_\ell)}) \cdot \xi(R(n, \ell))$, $\ell \in [M]$.
- $\xi(\mathbf{r}_n)$ be the row vector in which $\xi(\mathbf{r}_n)(m) = \xi(\mathbf{r}_n(m))$, $m \in [M]$; namely, it is the binary vector that identifies all items that u_n had rated.

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

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

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 other variants of those similarity scores and prediction formulas. We focus here on the version that is suggested in [11]. The modification 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 $h \geq 1$. Following the discussion in [11, Section 2.3], we implement herein the following ranking procedure. Let $J(n)$ be the subset of indices of items that u_n already rated. Define for each $m \in [M] \setminus J(n)$ the score

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

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. Then, the top h items to be recommended to u_n are those with the highest values of $\hat{s}(m)$.

3.2 Distributed scenarios

We consider distributed scenarios in which there are K vendors, $V_1, \ldots, V_K$, where each one of them offers a subset of items to some subset of users. Let $I_k \subseteq [N]$ denote the set of indices of users that V_k serves, $J_k \subseteq [M]$ denote the set of indices of the items that V_k offers, and $N_k := |J_k|$ and $M_k := |I_k|$ be their corresponding sizes, $k \in [K]$. The subsets I_k and J_k, $k \in [K]$, are assumed to be publicly known; namely, everybody knows that a given vendor offers, say, consumer electronic products and ships only to the bay area.[1]

[1] In our concluding remarks, Section 6, we discuss relaxations of this assumption.

The study of privacy-preserving collaborative filtering considered one of the following three distributed scenarios:

Horizontal: All vendors offer all items in B but they serve disjoint subsets of users from U. Namely, $J_k = [M]$ for all $k \in [K]$, but the sets $I_1, \ldots, I_K$ are all disjoint and their union is $[N]$. In particular, if R is the $N \times M$ user-item matrix, then V_k owns the subset of R's rows that corresponds to I_k, $k \in [K]$.

Vertical: All vendors serve all users in U but they offer disjoint subsets of items from B. Namely, $I_k = [N]$ for all $k \in [K]$, but the sets $J_1, \ldots, J_K$ are all disjoint and their union is $[M]$. Here, V_k owns the subset of R's columns that corresponds to J_k, $k \in [K]$.

Hybrid: In that model, the user-item matrix is distributed in a general manner between the vendors (not necessarily by rows or columns), so that each entry in the matrix is held by a single vendor.

In this study we consider a **general distribution scenario**. Like the hybrid scenario, each vendor owns some sub-matrix, but the sub-matrices could be overlapping. Namely, it is possible that a user u_n who wishes to purchase an item b_m could do so through more than just one vendor, as is the underlying assumption in all above described scenarios. That is the typical case in real markets – users usually have a choice between different vendors. Hence, the general distribution scenario is more realistic than those considered so far in prior art, and it includes all of them as private cases.

For simplicity we will assume that a user who purchased and rated an item through one vendor, had not purchased or rated that item again through another vendor. This natural assumption, which was not needed in previous studies who considered distribution scenarios with no overlaps, is made only for the sake of simplicity. However, it is possible that Alice, who had purchased a book and liked it very much, decided later on to purchase another copy of that book also for her niece's birthday, but this time she made the purchase through another vendor that had put that book on sale. We note that even in such unusual cases of repeated ratings, the outputs of our protocols are still justifiable, without needing to modify the protocols. We defer the discussion of this point to Appendix A.

3.3 The mediated setting

We assume $D > 1$ independent mediators, T_d, $d \in [D]$, that assist the vendors in performing the computations. They are assumed to be semi-honest, i.e., they follow the prescribed protocols, but try to glean from the messages received during those protocols information on the user-item rating matrix. We let $\mathbf{T} := \{T_1, \ldots, T_D\}$ denote the set of all mediators.

Our working assumption is that of an *honest majority*: if a subset of $\mathbf{T}$ colludes and combines the information that they got, in order to extract information on private user-item rating data, the subset's size is less than $D/2$. Such an assumption is very common in the MPC literature, e.g. [8, 14, 25, 33]. In practice, it means that if a malicious adversary tries to "corrupt" mediators in attempt to get access to the information that goes through them, he will be able to corrupt less than half of those mediators.

3.4 Secret Sharing

Secret sharing [36] methods are protocols that enable to distribute a secret among a group of participants, such that each of them is allocated a piece of information, called *a share*, so that some subsets of those shares enable the reconstruction of the secret. In its most basic form, called *Threshold Secret Sharing*, the secret can be reconstructed only when a sufficient number of shares are combined together, while smaller sets of shares reveal no information at all on the secret. In our context, the secret holders will be the vendors, and the group of participants among whom the

secrets will be shared are the mediators. The domain of secrets is known in advance and it will be a finite field $\mathbb{Z}_p$, where p is some sufficiently large prime (in particular, larger than the set of participants as well as the number of possible secrets).

We will use the Shamir threshold secret sharing scheme [36]. It is a D'-out-of-D scheme in the following sense: if D' is an integer in the range $1 \leq D' < D$, then the shares that are distributed in that scheme allow the recovery of the secret x from *any* subset of D' shares, while any subset of $D' - 1$ shares reveals no information on the secret. The scheme has two procedures: Share and Reconstruct:

• $\mathsf{Share}_{D',D}(x)$. The procedure samples a uniformly random polynomial $g(\cdot)$ over $\mathbb{Z}_p$, of degree at most $D' - 1$, where the free coefficient is x. That is, $g(t) = x + \alpha_1 t + \alpha_2 t^2 + \ldots + \alpha_{D'-1} t^{D'-1}$, where α_j, $1 \leq j \leq D' - 1$, are selected uniformly at random from $\mathbb{Z}_p$. The procedure outputs D values – $g(1), \ldots, g(D)$ – where $x_d = g(d)$ is the share given to T_d, $d \in [D]$. It is easy to see that any selection of $D' - 1$ shares reveals nothing about the secret x, whereas any subset of D' shares fully determines x, by means of polynomial interpolation, as we describe next.

• $\mathsf{Reconstruct}_{D'}(x_1, \ldots, x_D)$. The procedure is given any selection of D' shares out of $\{x_1, \ldots, x_D\}$, and it then interpolates a polynomial $g(\cdot)$ of degree at most $D' - 1$ using the given points. It then outputs $x = g(0)$.

In what follows we apply secret sharing on secret vectors and matrices. By that we mean that when a vendor wishes to share a vector, or a matrix, among $\mathbf{T}$, he will compute and distribute shares in each component of the vector or matrix, independently. Consequently, any sufficiently large subset of the mediators will be able to reconstruct the shared vector or matrix by reconstructing each component independently. We also use D'-out-of-D secret sharing with $D' = \lfloor (D+1)/2 \rfloor$. With such threshold, the number of shares needed to reconstruct the secret is at least $D' \geq D/2$. Hence, under our assumption of honest majority (in the sense that if a collusion between some of the mediators occurs, it involves less than half of them), the shared secrets will remain fully protected.

We conclude this section by a note on the selection of the field's size p. Selecting the prime p to be a Mersenne prime (a prime of the form $p = 2^t - 1$ for some integer $t > 1$) is advantageous since multiplication of two field elements in such cases can be done without performing an expensive division (in case the multiplication result exceeds the modulus). We use herein the Mersenne prime $p := 2^{31} - 1$ since it is sufficiently large for our purposes, in the sense that whenever we perform secret sharing, the value of the shared secret is strictly smaller than p.

4 PRIVACY PRESERVING PROTOCOLS

In this section we present our privacy-preserving protocols. In Section 4.1 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 mediators $\mathbf{T}$. Then, we describe the online phase in which a given vendor V_k submits queries to $\mathbf{T}$ towards computing the predicted rating of some user u_n for an item b_m, $P(u_n, b_m)$, Eq. (2) (Section 4.2), or getting the top h items for a given user u_n (Section 4.3). The online phase is carried out solely by the relevant vendor, V_k, and the mediators, $\mathbf{T}$. Namely, the participation of all vendors is required only in the offline and less frequent phase. Finally, in Section 4.4 we discuss a more efficient execution of the offline computations, which relies on the (plausible) assumption that between every two invocations of the offline phase, only a small fraction of the entries in the user-item rating matrix had changed.

4.1 Offline model construction

Recall that R is the global user-item rating matrix; for every $n \in [N]$ and $m \in [M]$, $R(n,m)$ is the rating that u_n gave to b_m, or zero if no such rating was given. Let $\mathrm{sq}(R)$ and $\xi(R)$ be matrices of the same dimensions as R, whose entries are as follows:

$$\mathrm{sq}(R)(n,m) = (R(n,m))^2\,, \text{ and } \xi(R)(n,m) = \xi(R(n,m))\,,\ (n,m) \in [N] \times [M]\,.$$

The columns of R were denoted by $\mathbf{c}_m$, $m \in [M]$. Therefore, the columns of $\mathrm{sq}(R)$ and $\xi(R)$ are $\mathbf{c}_m^2$ and $\xi(\mathbf{c}_m)$, under our notation agreements (see Section 3.1). Hence, the similarity score between a given pair of items, say b_ℓ and b_m, $\ell < m \in [M]$, is given by $S(\ell,m) = z_1/\sqrt{z_2 z_3}$, where

$$z_1 = \langle \mathbf{c}_\ell, \mathbf{c}_m \rangle\,,\ z_2 = \|\mathbf{c}_{\ell|m}\|^2 = \langle \mathbf{c}_\ell^2, \xi(\mathbf{c}_m) \rangle\,,\ z_3 = \|\mathbf{c}_{m|\ell}\|^2 = \langle \xi(\mathbf{c}_\ell), \mathbf{c}_m^2 \rangle \tag{4}$$

(see Definition 3.1).

The user-item matrix in our case is distributed among K parties, the vendors. The vendor V_k, $k \in [K]$, possesses an $N_k \times M_k$ user-item rating matrix, denoted R_k. That matrix holds an entry for each user that V_k serves and for each item that V_k offers. As stated in Section 3.2, we assume that each nonzero entry in R occurs in just a single R_k, $k \in [K]$. Hence, R_k is the $N_k \times M_k$ sub-matrix of the global $N \times M$ matrix R, which corresponds to R's entries whose indices are in the Cartesian product $I_k \times J_k$. Stated differently,

$$R = \sum_{k \in [K]} [\![R_k]\!] \tag{5}$$

where $[\![R_k]\!]$ is the $(N \times M)$-"inflation" of R_k in the sense that

$$[\![R_k]\!]\,(i,j) = \begin{cases} R_k(i,j) & \text{if } (i,j) \in I_k \times J_k \\ 0 & \text{if } (i,j) \in [N] \times [M] \setminus I_k \times J_k \end{cases}\,.$$

Example. Let us consider a setting with $N = 5$ users, $M = 6$ items, and $K = 4$ vendors:

- V_1 serves users u_1, u_2, u_3 and offers items b_1, b_2, b_3, b_4.
- V_2 serves users u_3, u_4, u_5 and offers items b_1, b_4, b_5.
- V_3 serves users u_1, u_2, u_5 and offers items b_2, b_3, b_5, b_6.
- V_4 serves users u_4, u_5 and offers items b_2, b_6.

Eq. (6) illustrates the four sub-matrices R_k, $k \in [4]$, within their corresponding inflated versions, $[\![R_k]\!]$. The entries that are marked by a central dot correspond to entries in the user-item matrix that the corresponding vendor does not possess. The global user-item matrix, R, is shown in Eq. (7), and it is the sum of the $K = 4$ inflated matrices in Eq. (6). Note that, in agreement with our assumption, a nonzero entry occurs in only one of those matrices.

$$[\![R_1]\!] = \begin{pmatrix} 0 & 2 & 4 & 0 & \cdot & \cdot \\ 0 & 0 & 0 & 4 & \cdot & \cdot \\ 5 & 0 & 0 & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \qquad [\![R_2]\!] = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & 0 & 2 & \cdot \\ 0 & \cdot & \cdot & 2 & 0 & \cdot \\ 0 & \cdot & \cdot & 3 & 0 & \cdot \end{pmatrix}$$

$$[\![R_3]\!] = \begin{pmatrix} \cdot & 0 & 0 & \cdot & 0 & 2 \\ \cdot & 0 & 0 & \cdot & 1 & 4 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 5 & 0 & \cdot & 1 & 0 \end{pmatrix} \qquad [\![R_4]\!] = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 3 & \cdot & \cdot & \cdot & 0 \\ \cdot & 0 & \cdot & \cdot & \cdot & 1 \end{pmatrix} \tag{6}$$

$$R = \sum_{k=1}^{4} [[R_k]] = \begin{pmatrix} 0 & 2 & 4 & 0 & 0 & 2 \\ 0 & 0 & 0 & 4 & 1 & 4 \\ 5 & 0 & 0 & 1 & 2 & 0 \\ 0 & 3 & 0 & 2 & 0 & 0 \\ 0 & 5 & 0 & 3 & 1 & 1 \end{pmatrix} \tag{7}$$

Protocol 1 is executed by the vendors and the mediators, **T**, towards the goal of **T** learning the similarity scores between items, Eq. (1). The input to the protocol is the user-item sub-matrices R_k, $k \in [K]$, that are held by the vendors. The protocol has three phases:

(1) Phase 1: The vendors distribute shares relating to their sub-matrices of user-item ratings.
(2) Phase 2: The mediators compute shares relating to the global user-item matrix.
(3) Phase 3: The mediators compute the similarity scores between every pair of items.

In Phase 1 (Lines 1-5) each vendor, V_k, $k \in [K]$, first computes $\mathrm{sq}(R_k)$ and $\xi(R_k)$, where

$$\mathrm{sq}(R_k)(i,j) = (R_k(i,j))^2\,,\ \xi(R_k)(i,j) = \xi(R_k(i,j))\,,\ (i,j) \in I_k \times J_k\,. \tag{8}$$

As an example, we illustrate the matrices R_1, $\mathrm{sq}(R_1)$ and $\xi(R_1)$ corresponding to $[[R_1]]$ from Eq. (6):

$$R_1 = \begin{pmatrix} 0 & 2 & 4 & 0 \\ 0 & 0 & 0 & 4 \\ 5 & 0 & 0 & 1 \end{pmatrix},\ \mathrm{sq}(R_1) = \begin{pmatrix} 0 & 4 & 16 & 0 \\ 0 & 0 & 0 & 16 \\ 25 & 0 & 0 & 1 \end{pmatrix},\ \xi(R_1) = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

The vendor V_k then proceeds to generate D'-out-of-D shares in each entry of each of those three matrices, with $D' = \lfloor (D+1)/2 \rfloor$, and distributes those shares to the mediators.

In Phase 2 (Lines 6-9) each mediator accumulates the shares received from all vendors. To that end, T_d (for every $d \in [D]$) first initializes three share matrices, denoted R^d, $\mathrm{sq}(R)^d$ and $\xi(R)^d$, to be $N \times M$ zero matrices. Then, whenever he gets sub-matrices of shares from a vendor, V_k, he adds those shares to the relevant entries of the matrix (R^d, $\mathrm{sq}(R)^d$ or $\xi(R)^d$). In view of Eq. (5) and the linearity of secret sharing, in the end of this phase the collection of all D shares $\{R^d(n,m)\}_{d\in[D]}$ are D'-out-of-D shares in $R(n,m)$ for every $(n,m) \in [N] \times [M]$. Similarly, the entries of the matrices $\mathrm{sq}(R)^d$ and $\xi(R)^d$, $d \in [D]$, are D'-out-of-D shares in the entries of $\mathrm{sq}(R)$ and $\xi(R)$, respectively. (Note that here we rely on our assumption that each user had rated each item at most once, see Section 3.2. In Appendix A we discuss the case in which a user rated some item more than once.)

Finally, in Phase 3 (Lines 10-18), the mediators use those shares in order to compute the similarity score between every pair of items, b_ℓ and b_m, $1 \le \ell < m \le M$. As explained earlier, this computation is carried out by computing the three inner products z_1, z_2 and z_3 in Eq. (4). This is done in Lines 11-13 in Protocol 1, by invoking Protocol 2, which we explain later on. Then, they compute from those three values the final score $S(\ell, m)$, according to Definition 3.1 (Lines 14-17). Since the similarity scores will be used later on in computations over $\mathbb{Z}_p$, the mediators translate the real-valued scores into integral ones by multiplying them by a sufficiently large factor Q and rounding to the nearest integer. We used in our experiments $Q = 1000$ (a choice that preserves an accuracy of three digits after the decimal point). Finally, as the similarity score matrix S is symmetric, the mediators store the similarity score $S(\ell, m)$ which they just computed also in $S(m, \ell)$ (Line 18).

We note that the computation of the similarity score between a given pair of items can be done by just one of the mediators and then broadcast to all other mediators. Hence, for efficiency, the mediators may split among themselves the entries of the similarity matrix so that the final computations in each such entry is carried out by only one of the mediators.

Protocol 2 computes inner products between vectors in $\mathbb{Z}_p^N$ which are shared by a D'-out-of-D secret sharing scheme among the D mediators, where $D' = \lfloor (D+1)/2 \rfloor$. It serves as a sub-protocol

Protocol 1: Computing the similarity matrix.

Input: Each V_k, $k \in [K]$, holds a matrix $\{R_k(i,j) : (i,j) \in I_k \times J_k\}$.

1 **forall** $k \in [K]$ **do**
2 V_k computes $\mathrm{sq}(R_k)$ and $\xi(R_k)$.
3 V_k computes D'-out-of-D shares, $\{R_k^d(i,j) : (i,j) \in I_k \times J_k\}_{d\in[D]}$, in each entry in R_k.
4 V_k sends the dth set of shares to T_d, for all $d \in [D]$.
5 V_k does similarly with the matrices $\mathrm{sq}(R_k)$ and $\xi(R_k)$.
6 **forall** $d \in [D]$ **do**
7 T_d initializes $N \times M$ matrices, denoted R^d, $\mathrm{sq}(R)^d$ and $\xi(R)^d$, to be the zero matrices.
8 When receiving from V_k the set of shares $\{R_k^d(i,j) : (i,j) \in I_k \times J_k\}$, T_d updates the entries of the matrix R^d as follows: $R^d(i,j) \leftarrow R^d(i,j) + R_k^d(i,j)$ for all $(i,j) \in I_k \times J_k$.
9 Similarly, T_d updates the entries of the matrices $\mathrm{sq}(R)^d$ and $\xi(R)^d$.
10 **forall** $1 \le \ell < m \le M$ **do**
11 The mediators invoke Protocol 2 to compute $z_1 = \langle \mathbf{c}_\ell, \mathbf{c}_m \rangle$ from their shares in $\mathbf{c}_\ell$ and $\mathbf{c}_m$.
12 Similarly, they compute $z_2 = \langle \mathbf{c}_\ell^2, \xi(\mathbf{c}_m) \rangle$ from their shares in $\mathbf{c}_\ell^2$ and $\xi(\mathbf{c}_m)$.
13 Similarly, they compute $z_3 = \langle \xi(\mathbf{c}_\ell), \mathbf{c}_m^2 \rangle$ from their shares in $\xi(\mathbf{c}_\ell)$ and $\mathbf{c}_m^2$.
14 **if** $z_2 z_3 \neq 0$ **then**
15 The mediators set $S(\ell,m) \leftarrow \lfloor Q z_1 / \sqrt{z_2 z_3} + 0.5 \rfloor$.
16 **else**
17 The mediators set $S(\ell,m) = 0$.
18 $S(m,\ell) \leftarrow S(\ell,m)$.

Output: **T** get $S(\ell,m)$.

in Lines 11-13 in Phase 3 of Protocol 1. Note that the goal of Phases 1 and 2 in Protocol 1 is to ensure the initial requirement in Protocol 2.

Let $\mathbf{v}_1$ and $\mathbf{v}_2$ be the two secret vectors in $\mathbb{Z}_p^N$, in which the mediators hold shares. The goal is to compute $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle$. For each $i \in [2]$ and each $n \in [N]$, the D values $\{\mathbf{v}_{i,d}(n)\}_{d\in[D]}$ are D'-out-of-D shares in $\mathbf{v}_i(n)$. The meaning of that is that there exists a polynomial $f_{i,n}$ of degree $D'-1$ over $\mathbb{Z}_p$, such that the share $\mathbf{v}_{i,d}(n)$ that T_d holds in $\mathbf{v}_i(n)$ equals $f_{i,n}(d)$, $d \in [D]$. Now, consider the polynomial $F := \sum_{n=1}^N f_{1,n} \cdot f_{2,n}$. It is a polynomial of degree $2(D'-1)$ over $\mathbb{Z}_p$. In particular, one needs $2D'-1$ point values in F in order to reconstruct it. In addition, $F(0) = \sum_{n=1}^N f_{1,n}(0) \cdot f_{2,n}(0) = \sum_{n=1}^N \mathbf{v}_1(n) \cdot \mathbf{v}_2(n) = \langle \mathbf{v}_1, \mathbf{v}_2 \rangle$. The value that each T_d computes in Line 1 equals $s_d = \sum_{n\in[N]} \mathbf{v}_{1,d}(n) \cdot \mathbf{v}_{2,d}(n) = \sum_{n\in[N]} f_{1,n}(d) \cdot f_{2,n}(d) = F(d)$. Therefore, the set of shares $\{s_d\}_{d\in[D]}$ is a set of $(2D'-1)$-out-of-D shares in the desired inner product. Our setting of D' ensures that $2D'-1 \le D$. (Specifically, if D is odd then $2D'-1 = D$, while if D is even then $2D'-1 = D-1$.) Hence, any selection of $2D'-1$ mediators can use their s_d-shares that were computed in Line 1 in order to interpolate F (Line 2) and recover $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle$ (Line 3).

4.1.1 Computational and communication costs. The computational cost for vendor V_k, $k \in [K]$, is $O(|I_k| \cdot |J_K| \cdot D)$ (Lines 1-5 in Protocol 1), which is bounded by $O(NMD)$. The cost of Phase 2 (Lines 6-9) for each mediator is bounded by $O(NMK)$, while the cost of Phase 3 is bounded by $O(M^2(N+D))$. As for communication, each vendor V_k, $k \in [K]$, has to send to each of the D mediators three shares relating to each of the $|I_k| \cdot |J_k|$ entries in his sub-matrix. Hence, the communication cost is $\sum_{k\in[K]} 3D\lceil \log p \rceil \cdot |I_k| \cdot |J_k|$ bits (which, in the vertical, horizontal or hybrid distribution scenarios equals $3NMD\lceil \log p \rceil$).

Protocol 2: InnerProduct: Computing inner product between shared vectors.

Input: $\mathbf{v}_i$, $i \in [2]$, are two vectors in $\mathbb{Z}_p^N$. Every mediator T_d in $\mathbf{T} = \{T_1, \ldots, T_D\}$ holds D'-out-of-D vector shares in them, denoted $\mathbf{v}_{i,d}$.

1 Each T_d, $d \in [D]$, computes $s_d \leftarrow \sum_{n\in[N]} \mathbf{v}_{1,d}(n) \cdot \mathbf{v}_{2,d}(n)$.

2 Any selection of $2D' - 1$ mediators out of $\mathbf{T}$ use their s_d-shares in order to interpolate a polynomial $F(\cdot)$ of degree $2D' - 2$ that agrees with those $2D' - 1$ s_d-shares.

3 The inner product between the two input vectors is $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle \leftarrow F(0)$.

Output: $\mathbf{T}$ gets $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle$.

4.1.2 Privacy. The protocols maintain the privacy of the inputs provided by the vendors as those inputs are communicated to the mediators by D'-out-of-D secret sharing. Indeed, in order to recover the private user-item ratings, at least $D' = \lfloor (D+1)/2 \rfloor$ mediators would need to collaborate. But as we assumed that the mediators have an honest majority, then if some of them collude in order to try and reconstruct the private data, the number of such colluding mediators would be strictly less than $D/2 \le D'$; such a coalition of mediators would not be able to learn a thing on the shared private data.

We would like to stress that this shield of protection is in the *information-theoretic* sense: namely, even if the mediators have *unlimited* computing power, they would not be to reveal any information on the shared user-item ratings. This level of security is higher than that offered by encryption-based protocols, as the latter rely on the assumption that the adversary (the mediators in our case) can execute only polynomial-time computations on the data that they received in order to extract information on the private inputs.

Note that our protocols are not perfectly secure, in the MPC sense, as they reveal to the mediators not only the final desired outputs (the similarity scores $S(\ell, m)$) but also the intermediate values z_1, z_2, z_3. Hiding even those intermediate values (which do not reveal private information about specific ratings of individual users) is possible, but that would significantly increase the runtimes of the protocols (see more on that in our concluding remarks, Section 6).

It is important to note that while in theory perfect privacy can always be achieved (e.g., by implementing generic solutions such as Yao's garbled circuit construction [51]), in practice privacy must always be balanced against efficiency. Therefore, 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. anonymization of distributed datasets [18, 42, 45, 53], distributed association rule mining [19, 35, 41, 48, 52], distributed constraint optimization problems [15–17, 23, 43, 44, 46, 47], distributed graph mining [3] and more. The same goes for all studies on PPCF studies that we reviewed in Section 2.

4.2 Computing predicted ratings

In view of Eq. (2) and the definitions preceding it, the predicted rating is given by

$$P(u_n, b_m) := \overline{R(b_m)} + \frac{u_{n,m} - v_{n,m}}{w_{n,m}}\,, \tag{9}$$

where

$$\begin{aligned} &\overline{R(b_m)} = \tfrac{\sum_{n\in[N]} R(n,m)}{\sum_{n\in[N]} \xi(R(n,m))}\,, && u_{n,m} := \textstyle\sum_{\ell\in[M]} \mathbf{s}_m(\ell) \cdot R(n,\ell)\,, \\ &v_{n,m} := \textstyle\sum_{\ell\in[M]} \mathbf{s}_m(\ell) \cdot \overline{R(b_\ell)} \cdot \xi(R(n,\ell))\,, && w_{n,m} := \textstyle\sum_{\ell\in[M]} \mathbf{s}_m(\ell) \cdot \xi(R(n,\ell))\,. \end{aligned} \tag{10}$$

Before moving on to explaining how the mediators can compute each of the values in Eq. (9), we observe that the average ratings, $\overline{R(b_\ell)}$, $\ell \in [M]$, are rational numbers. However, the computation of $v_{n,m}$, Eq. (10), must be carried out in the secret sharing field $\mathbb{Z}_p$, since it involves entries of the matrix $\xi(R)$, in which the mediators hold shares in $\mathbb{Z}_p$. Therefore, we replace the computation of $v_{n,m}$ in Eq. (10) with the following approximation,

$$v_{n,m} = \frac{1}{Q} \sum_{\ell \in [M]} c_\ell \cdot \xi(R(n,\ell)) \quad \text{where } c_\ell = \lfloor Q \cdot \mathbf{s}_m(\ell) \cdot \overline{R(b_\ell)} + 0.5 \rfloor \,. \tag{11}$$

Here, Q is a re-scaling factor like the one that we used in Line 15 of Protocol 1 in order to translate the read-valued similarity scores into integral ones.

4.2.1 Computing affine combinations of secrets. Before we describe the needed computations in the offline phase (Section 4.2.2) and in the online phase (Section 4.2.3), we observe that the sums in Eqs. (10) and (11) are all linear combinations of secrets. Hence, we make herein the following helpful observation:

Let r_j be secrets that are shared among the mediators by a D'-out-of-D secret sharing scheme. Assume that $\sigma = \beta + \sum_j \gamma_j r_j$, where β and γ_j are known integers. Then the mediators can compute the sum σ, without recovering the secrets r_j, as follows. Letting r_j^d denote T_d's share in r_j, $d \in [D]$, then by the linearity of secret sharing, the values $\{\beta + \sum_j \gamma_j r_j^d : d \in [D]\}$ are proper D'-out-of-D shares in $\sigma = \beta + \sum_j \gamma_j r_j$. Hence, any subset of D' mediators can use the latter shares in order to recover the desired sum σ.

4.2.2 Offline computations. Here we identify all of the values in Eqs. (9)–(11) that do not depend on n and, consequently, can be computed by the mediators already in the offline phase:

(1) Compute the numerator and denominator in the expression for $\overline{R(b_m)}$ in Eq. (10) from the shares in R's and $\xi(R)$'s entries, as explained in Section 4.2.1, and then divide them in order to obtain $\overline{R(b_m)}$ for all $m \in [M]$.
(2) Compute the vectors $\mathbf{s}_m$, $m \in [M]$, from the similarity score matrix S, according to their definition in Section 3.1.
(3) Given the average item ratings and the vectors $\mathbf{s}_m$, the mediators will proceed to compute the integer coefficients c_ℓ, $\ell \in [M]$, as defined in Eq. (11).

4.2.3 Online computations. Assume that a vendor V_k submitted a query in the form $(n, m) \in I_k \times J_k$. Namely, a query that asks for a predicted rating that a user u_n, whom V_k serves ($n \in I_k$), would give to an item b_m that V_k offers ($m \in J_k$). Upon receiving such a query, the mediators will compute $u_{n,m}$, $w_{n,m}$ and $v_{n,m}$ from the shares that they got in R's and $\xi(R)$'s entries, and the known coefficients in the linear combinations in Eqs. (10)+(11), respectively, as described in Section 4.2.1. Then, they will compute the predicted rating, $P(u_n, b_m)$, by plugging into Eq. (9) the values $u_{n,m}$, $w_{n,m}$ and $v_{n,m}$ which they had just computed, as well as the item average rating $\overline{R(b_m)}$, which they had already computed in the offline phase. Finally, they will send the computed predicted rating to V_k.

Throughout the above described process, the private user-item ratings remain protected against the mediators, under our assumption of honest majority.

4.3 Computing the most recommended items

When a vendor V_k, $k \in [K]$, wants to recommend to one of his users, u_n, $n \in I_k$, items that will most likely appeal to her, V_k submits a query to the mediators so that they can jointly and privately find the h items from $\{b_m : m \in J_k\}$ that maximize the score $\hat{s}(m)$, Eq. (3), and that u_n still had not rated. Define for every $m \in [M]$ the vector $\mathbf{s}'_m$ as follows: $\mathbf{s}'_m(\ell) = S(m,\ell)$ if $\ell \in N_q(m)$ while

$\mathbf{s}'_m(\ell) = 0$ otherwise.[2] Then, by Eq. (3),

$$\hat{s}(m) = \sum_{\ell \in [M]} \mathbf{s}'_m(\ell) \cdot \xi(R(n,\ell)) \,. \tag{12}$$

The mediators will compute the vectors $\mathbf{s}'_m$, $m \in [M]$, once in the offline phase. The vectors $\mathbf{s}'_m$ may have negative entries, since the set of q nearest neighbors of b_m could include items b_ℓ for which $S(m,\ell) < 0$. We note that $S(m,\ell) \in [-Q, Q]$. Indeed, the original cosine-score, Eq. (1), is confined to the interval $[-1, 1]$, as implied by the Cauchy-Schwarz inequality. Hence, after the rescaling by Q (see Line 15 in Protocol 1), the entries of the matrix S are confined to the interval $[-Q, Q]$. Therefore, as the number of nonzero addends in the sum in Eq. (12) is at most q, we infer that $\tilde{s}(m) := (qQ + 1) + \hat{s}(m) \geq 1$. As the mapping from $\hat{s}$ to $\tilde{s}$ is monotone, we can look for the h items that maximize $\tilde{s}$.

We proceed to present Protocol 3 that privately computes the subset of h items b_m, $m \in J_k$, with highest $\tilde{s}(m)$ scores. The protocol starts by V_k submitting a query to the mediators (Line 1). That query includes an index $n \in I_k$ of a user u_n that V_k serves. Then (Line 2), the mediators jointly generate a random and secret permutation π over J_k – the set of indices of items that V_k offers.[3]

Next, the mediators perform the computations in Lines 3-6. Each mediator T_d sets a vector of shares $\mathbf{x}_d$ which is initialized to hold in its mth entry, $m \in J_k$, the value $(qQ + 1) + \sum_{\ell \in [M]} \mathbf{s}'_m(\ell)\xi(R)^d(n,\ell)$ (Line 4). In view of Eq. (12) and our discussion in Section 4.2.1, the set $\{\mathbf{x}_d(m) : d \in [D]\}$ is a set of D'-out-of-D shares in $\tilde{s}(m) = (qQ + 1) + \hat{s}(m)$. Next (Line 5), T_d multiplies each entry in $\mathbf{x}_d$ with the corresponding D'-out-of-D share in $1 - \xi(R)(n,m)$. The resulting set $\{\mathbf{x}_d(m) : d \in [D]\}$ is now a set of $(2D' - 1)$-out-of-D shares in $\tilde{s}(m) \cdot (1 - \xi(R)(n,m))$ (as it is the same computation of producing shares in a product of two shared secrets, like we did in Protocol 2). Then (Line 6), T_d sends to V_k the vector $\mathbf{y}_d$ which is the result of permuting $\mathbf{x}_d$ entries using the secret permutation π.

After collecting the responses from all mediators, V_k selects a subset of $2D' - 1$ vectors from $\{\mathbf{y}_d : d \in [D]\}$ in order to interpolate their entries for each $m \in J_k$ and recover the underlying shared secret, denoted z_m (Line 7). In view of the above discussion, $z_m = \tilde{s}(\pi^{-1}(m)) \cdot (1 - \xi(R)(n, \pi^{-1}(m)))$. Namely, z_m equals the $\tilde{s}$-score of the item $b_{\pi^{-1}(m)}$, if that item had not been rated so far by u_n, while z_m equals zero otherwise. Since $\tilde{s}(m) \geq 1$ for all m, then the indices $i_1, \ldots, i_h$ of the largest values in $\{z_m : m \in J_k\}$ identify the indices of the h items that were not rated so far by u_n and have largest $\tilde{s}$-scores. V_k sends those indices to the mediators who proceed to apply on them the inverse permutation π^{-1} and send them back to V_k (Lines 8-9). Consequently, V_k now has the indices of the top h items, yet unrated by u_n, which maximize the $\tilde{s}$- and $\hat{s}$-scores.

4.3.1 Privacy. The only information that the mediators obtain in the course of Protocol 3 is the indices of the top h items to recommend to u_n. In case that such information may be deemed sensitive, it is possible to apply an additional layer of security as follows. The vendors agree upfront on a random and secret orderings of both the user set $U = \{u_1, \ldots, u_N\}$, and the global set of items, $B = \{b_1, \ldots, b_M\}$. If those random orderings are kept secret from the mediators, then Protocol 3 reveals no information to the mediators.

As for the vendor V_k, the protocol does leak to him more information than just the desired output, being the sought-after h indices. V_k learns also the set of $\tilde{s}$-scores of all items that were still unrated by u_n. However, owing to the random permutation π which is kept secret from him, V_k is unable to

[2] Note that the definition of $\mathbf{s}'_m$ slightly differs from that of $\mathbf{s}_m$, which is used in the context of predicted ratings, Section 4.2, as $\mathbf{s}'_m$ is defined by the index set $N_q(m)$ while $\mathbf{s}_m$ is defined by the index set $N_q^+(m)$, see Section 3.1.

[3] The mediators may perform that computation using the Fisher–Yates shuffle algorithm (see e.g. [22]), where the random numbers can be generated by jointly creating a random seed and feeding it into a linear-feedback shift register, such as the Well Equi-distributed Long-period Linear (WELL) family of pseudorandom number generators [31].

Protocol 3: Computing for u_n the top h yet unrated items offered by V_k.

1 V_k submits to $\mathbf{T}$ a query $n \in I_k$.
2 $\mathbf{T}$ jointly generate a secret and random permutation π over J_k.
3 **forall** $d \in [D]$ **do**
4 $\quad T_d$ defines an M_k-dimensional vector $\mathbf{x}_d$ and sets $\mathbf{x}_d(m) \leftarrow (qQ+1) + \sum_{\ell \in [M]} \mathbf{s}'_m(\ell)\xi(R)^d(n,\ell)$, $\forall m \in J_k$.
5 $\quad T_d$ computes $\mathbf{x}_d(m) \leftarrow \mathbf{x}_d(m) \cdot (1 - \xi(R)^d(n,m))$, $\forall m \in J_k$.
6 $\quad T_d$ sends to V_k the permuted vector $\mathbf{y}_d \leftarrow \pi(\mathbf{x}_d)$.
7 For every $m \in J_k$, V_k uses $2D'-1$ shares out of the received shares $\{\mathbf{y}_d(m) : d \in [D]\}$ in order to recover the underlying shared secret z_m, $m \in J_k$.
8 V_k identifies the h values of z_m, $m \in J_k$, which are largest, and sends their indices, denoted $\{i_1, \ldots, i_h\}$, to $\mathbf{T}$.
9 The mediators send back to V_k the set $\{\pi^{-1}(i_1), \ldots, \pi^{-1}(i_h)\}$.
Output: V_K gets the indices in $J_k \setminus J(n)$ of the top h items to be recommended to u_n.

associate any of those scores to any specific item (beyond the fact that the top h scores relate to the top h items). Such excess information is benign since it does not disclose private user-item ratings owned by other vendors.

A single malicious mediator may disclose to the vendor V_k the secret permutation π; consequently, V_k would learn the scores $\hat{s}(m)$, Eq. (3), of each of his items. Such excess information does not jeopardize the privacy of users or other vendors, just as predicted ratings (Section 4.2) do not. In fact, some recommender systems may require such information as they present to their users a personal "match score" for each of the most recommended items (that is the case e.g. in Netflix).

The above described benign leakage of information to a coalition of a mediator and a vendor could be eliminated if the mediators carry out the entire computation on their own, as we proceed to outline. The mediators will skip Line 2 in Protocol 3, as they would have no interaction with the vendor V_k, and thus there would be no need for such a secret permutation. They will compute $x_d(m)$ as described in Line 4 of Protocol 3; the set $\{x_d(m) : d \in [D]\}$ is a set of D'-out-of-D shares in the shifted score $\tilde{s}(m) = (qQ+1) + \hat{s}(m)$, $m \in J_k$. Then, instead of Line 5, the mediators will invoke an MPC sub-protocol (see e.g. [8, 10]) in order to compute from those shares, and from the D'-out-of-D shares that they hold in $1 - \xi(R)(n,m)$, a set of D'-out-of-D shares in the product $\tilde{s}(m) \cdot (1 - \xi(R)(n,m))$. (The difference with respect to the computation that takes place in Line 5 of Protocol 3 is that the latter computation results in $(2D'-1)$-out-of-D shares in the same product, instead of D'-out-of-D shares, as we need for the next computation.) Afterwards, the mediators can invoke a secure comparison sub-protocol [29] in order to identify the h items that maximize $\tilde{s}(m) \cdot (1 - \xi(R)(n,m))$ (namely, the h items that u_n had not rated yet and maximize $\hat{s}(m)$) without recovering those scores. In conclusion, the mediators will notify V_k of the indices of those items.

Such a course of action would entail higher runtimes. The information that such a protocol protects, in comparison with the previously described scenario in which a malicious mediator discloses the secret permutation in Protocol 3, is of a very benign nature, and in some recommender systems it is even part of the desired output. Hence, such an MPC-enhanced protocol might not be attractive in some practical recommendation systems.

4.4 Efficient execution of the offline computations in wake of updates in the user-item rating matrix

The computations in the offline phase (Section 4.1) have to be repeated in some periodicity in order to update the similarity matrix due to recent activities of the users (e.g., Alice added a rating to a movie she saw, or Bob decided to increase the rating he had previously given to some book). It is possible to run Protocol 1 every predetermined update period (say, once every day), in order to incorporate such recent updates in the similarity model. However, as the vast majority of the user-item matrix entries would remain the same, most of the computations would be redundant. Here we suggest a more careful implementation of Protocol 1, which takes into account that only a small fraction of the matrix entries had changed.

Assume that in the current update period, user u_{n_i} rated (or changed a previous rating for) item b_{m_i}, $1 \le i \le t$. Namely, t of R's entries, $\{R(n_i, m_i) : 1 \le i \le t\}$, need to be updated. One way to perform these updates is that the relevant vendors, through whom those activities took place, would send to the mediators updated shares only for those entries, instead of doing so for all of the entries in their corresponding sub-matrices, as is done in Lines 1-5 of Protocol 1. Alas, that would disclose to the mediators which user rated what item. Even though the actual rating would be still hidden from the mediators, such an information leakage is prohibited as it violates the users' right for privacy. In contrast, the vendors could completely ignore efficiency and simply execute Lines 1-5 of Protocol 1 fully; that approach would maintain perfect privacy.

However, instead of those two extreme approaches, the first that is geared towards efficiency, and the second that is geared towards privacy, the vendors could take an intermediate approach. Recall that each vendor V_k possesses a sub-matrix R_k of dimensions $N_k \times M_k$ which relates to the entries $(n, m) \in I_k \times J_k$ in the global user-item rating matrix R that he controls. Assume that in a given update period, the subset of V_k's entries that had changed is $A_k \subseteq I_k \times J_k$. Then instead of updating just A_k (first approach) or all of $I_k \times J_k$ (second approach), V_k chooses a random subset $\overline{A}_k$ where $A_k \subseteq \overline{A}_k \subseteq I_k \times J_k$, and then he updates all entries in $\overline{A}_k$. By that we mean that V_k shares with the mediators the difference $R'_k(n, m) - R_k(n, m)$ for all $(n, m) \in \overline{A}_k$, where $R_k(n, m)$ is the previous value of that entry and $R'_k(n, m)$ is the updated value. Note that for all entries in $\overline{A}_k \setminus A_k$, that difference would be zero. As for the entries in A_k, in the typical case where this is the first time in which u_n rates item b_m, we will have $R'_k(n, m) - R_k(n, m) = R'_k(n, m)$. However, in cases where u_n modifies his rating to b_m, the value that V_k shares among the mediators is the difference between the previous and updated ratings.

Higher values of the ratio $|\overline{A}_k|/|A_k|$ imply higher levels of privacy, since the mediators can infer for any given entry in $\overline{A}_k$ that it was an entry that actually changed in probability $|A_k|/|\overline{A}_k|$ (under the assumption that the ratio $|\overline{A}_k|/|A_k|$ is known). On the other hand, the runtime for V_k in Lines 2-5 of Protocol 1, as well as the runtime for the mediators in Lines 8-9, will reduce by a factor of $N_k M_k/|\overline{A}_k|$.

As we shall see in Section 5, the main computational toll in all of Protocol 1 is the computation of inner products in Line 1 of Protocol 2 (that is called in Lines 11-13 of Protocol 1). Let us focus on a pair of items, $b_\ell, b_m \in B$. Let $\mathbf{c}_\ell$ and $\mathbf{c}_m$ denote their corresponding columns in R and let $\Delta\mathbf{c}_\ell$ and $\Delta\mathbf{c}_m$ denote the updates to those two columns. Then the mediators need to compute in Line 11 the inner product $\langle \mathbf{c}_\ell + \Delta\mathbf{c}_\ell, \mathbf{c}_m + \Delta\mathbf{c}_m \rangle$. By bilinearity,

$$\langle \mathbf{c}_\ell + \Delta\mathbf{c}_\ell, \mathbf{c}_m + \Delta\mathbf{c}_m \rangle = \langle \mathbf{c}_\ell, \mathbf{c}_m \rangle + \langle \mathbf{c}_\ell, \Delta\mathbf{c}_m \rangle + \langle \Delta\mathbf{c}_\ell, \mathbf{c}_m \rangle + \langle \Delta\mathbf{c}_\ell, \Delta\mathbf{c}_m \rangle \,. \tag{13}$$

The first inner product on the right hand side of Eq. (13) is the current inner product that was already computed in the last update period. Hence, it suffices to compute each of the other three inner products on the right hand side of Eq. (13). But since those are inner products between vectors

dataset	N	M	$numR$	density	scale
ML100K	943	1682	10^5	6.30%	[1,5]
ML1M	6040	3706	10^6	4.47%	[1,5]
ML10M	71567	10677	10^7	1.30%	[1,5]
ML20M	138493	26744	$2 \cdot 10^7$	0.54%	[1,5]
WN0	100000	10000	$2 \cdot 10^7$	2%	[1,5]
WN1	250000	10000	$5 \cdot 10^7$	2%	[1,5]
WN2	500000	10000	$1 \cdot 10^8$	2%	[1,5]
WN3	1000000	10000	$2 \cdot 10^8$	2%	[1,5]
WM0	100000	10000	$2 \cdot 10^7$	2%	[1,5]
WM1	100000	25000	$5 \cdot 10^7$	2%	[1,5]
WM2	100000	50000	$1 \cdot 10^8$	2%	[1,5]
WM3	100000	100000	$2 \cdot 10^8$	2%	[1,5]

Table 1. Dataset characteristics

of dimensions that are significantly smaller than N (because both $\Delta\mathbf{c}_\ell$ and $\Delta\mathbf{c}_\ell$ are vectors that involve only a small fraction of the users), the time to compute them is significantly smaller than the time that is required to compute an inner product of two vectors of dimension N. In the next section we illustrate the advantages of this approach in several simulated scenarios.

5 EXPERIMENTS

5.1 overview

Our protocols issue the very same similarity scores, predicted ratings and predicted rankings as described in Section 3.1; namely, the privacy-preserving mechanisms that we apply do not alter the computed outputs (as opposed to obfuscation- or clustering-based methods). In particular, they issue the very same outputs as the protocols of [38]. Hence, we focus herein on runtime experiments in order to show the dramatic advantage that our protocols offer, in comparison to [38], in terms of runtimes. We stress that apart from runtime, our protocols offer other significant advantages, of qualitative nature, as discussed in the Introduction.

▹ Experimental setting. All experiments were run on a virtual machine in the Google Could Platform with the c2-standard-60 machine (60 vCPUs, 240 GB memory). The algorithms were implemented in C++.

▹ Datasets. We used four publicly available datasets: MovieLens 100K, MovieLens 1M, MovieLens 10M, and MovieLens 20M. (In all tables hereinafter we refer to those datasets by the following abbreviations: ML100K, ML1M, ML10M, and ML20M.) Those datasets are available online as CSV files at https://grouplens.org/datasets/movielens/X, where the suffix 'X' is '100K', '1M', '10M', or '20M'. In addition, we generated artificial datasets in order to test the scalability of our algorithms with respect to the number of users, N, and number of items, M.[4] The datasets for testing the runtime dependency on N (M) are denoted WNi (WMi), where $i = 0, 1, 2, 3$. Table 1 reports the main characteristics of all datasets: number of users N, number of items M, number of ratings $numR$, density (in percents) — $\frac{numR}{NM} \times 100$, and the rating scale.

[4]The content of those datasets was generated as follows: 2 percents of their entries were filled with random ratings between 1 and 5, while the remaining entries were set to zero. Having said that, We stress that the runtime of our algorithms is not affected by the content of the underlying dataset.

We measured the runtimes for the vendors and the mediators to perform the computations in the offline and online phases, and report them in Sections 5.2 and 5.3, respectively.

5.2 The offline phase

The main computational effort is in the offline phase. The computations that the vendors need to perform are in Lines 2-5 of Protocol 1. As for the mediators, their offline tasks consist of (a) Lines 7-9 in Protocol 1 for aggregating the share inputs from all vendors; (b) the main loop in Lines 10-18 of Protocol 1 for computing the similarity scores between all pairs of items; (c) the offline computations as described in Section 4.2.2; (d) the offline computations as described in Section 4.3 (i.e., computing the vectors $\mathbf{s}'_m$, $m \in [M]$).

5.2.1 Runtimes for the vendors. The runtimes for the vendors are shown in Table 2. We assumed that there exists only one vendor who holds the entire user-item matrix. In a distributed setting, where the user-item matrix is distributed between several vendors (see Section 3.2), one may derive the runtimes for a vendor V_k by multiplying the runtimes shown in Table 2 by the proportion of the matrix that is held by V_k, namely, by the fraction $\frac{N_k M_k}{NM}$. So for example, if the dataset MovieLens 20M is split evenly between $K = 5$ vendors, then each of the vendors will spend around 70 seconds in the case $D = 3$ and up to 4.13 minutes in the case $D = 9$. Such overhead is negligible as it occurs very infrequently (say, once every few days) and can be carried out in idle time.

We note that larger values of D increase the runtime for the vendors, since they need to utilize polynomials of higher degree and generate more shares, for each of the entries in the matrices R_k, $\mathrm{sq}(R_k)$ and $\xi(R_k)$.

We proceed to compare those results to the corresponding runtimes of the Single-Mediator Protocols (SMP) of [38]. We focus here on their protocol for the vertical scenario. In that scenario, each vendor V_k who possesses M_k columns out of the M columns in the matrix, has to perform $2NM_k$ encryptions. The right side of Table 2 shows the corresponding runtimes, when the homomorphic cipher is Paillier [30] (as used in [38]) with a modulus of 512 or 1024 bits. As with the runtimes for our distributed mediation protocols, we assumed that there is a single vendor, namely, the shown runtimes are the times to compute $2NM$ encryptions. In a genuine vertical split, those runtimes should be multiplied by the fraction of columns owned by that vendor (namely, by M_k/M). So for example, when there are K vendors and each of them possesses $M_k \sim M/K$ columns, the shown runtimes should be divided by K. As can be seen, the Single-Mediator protocol is not scalable. For example, in a balanced vertical split with $K = 5$, the runtime for each vendor with MovieLens 10M is roughly 24 hours, when using 512-bit encryptions, and more than a week when using 1024-cryptography. The runtimes for MovieLens 20M are roughly five times worse. However, as we already saw, our protocol's runtime on the MovieLens 20M dataset is only 4.13 minutes per vendor in the $K = 5$ vertical split scenario and $D = 9$ mediators.

For both our protocol and the Single-Mediator protocol, the vendor's runtime depends linearly on the number of entries in the dataset, because the main effort in both protocols is in the cryptographic processing of each of the dataset entries (secret sharing in our protocol and encryption in the Single-Mediator protocol).

5.2.2 Runtimes for the mediators. The runtimes for the mediators are shown in Table 3, on the left side of each column. The percentage shown on the right side of each column relates to the time spent in Line 1 of Protocol 2. For example, in the 20M dataset, when $D = 9$, nearly 54 minutes are spent in Line 1 of Protocol 2, while less than half a minute is spent on all other offline computations.

As can be seen, in the larger datasets, the runtimes for the mediators are much larger than those for the vendors. However, the lion's share of those runtimes is spent on computing the

dataset	D=3	D=5	D=7	D=9	SMP_{512}	SMP_{1024}
ML100K	0.095	0.224	0.296	0.415	0.898×10^3	6.62×10^3
ML1M	1.881	3.526	4.567	7.259	1.267×10^4	9.343×10^4
ML10M	72.925	110.062	184.564	220.324	4.327×10^5	31.894×10^5
ML20M	349.326	647.116	904.904	1240.474	2.097×10^6	15.459×10^6
WN0	73.025	132.725	184.625	254.925	5.664×10^5	41.74×10^5
WN1	182.187	331.437	461.937	636.937	1.996×10^5	14.714×10^5
WN2	365.25	663.75	923.25	1275.875	3.993×10^5	29.428×10^5
WN3	733.75	1330.75	1852.75	2552.25	5.664×10^6	41.74×10^6
WM0	73.025	132.725	184.625	254.925	5.664×10^5	41.74×10^5
WM1	187.687	339.187	480.187	636.937	1.996×10^5	14.714×10^5
WM2	375.875	678.875	960.875	1275.875	3.993×10^5	29.428×10^5
WM3	755.25	1364.25	1922.25	2552.25	5.664×10^6	41.74×10^6

Table 2. Runtimes (seconds) for a vendor in the offline phase. Left: our distributed mediated protocol for different values of D; right: runtimes for the Single-Mediator Protocols (SMP) of [38] with 512- and 1024-bit encryptions, in the vertical distribution scenario. All runtimes in the table need to be multiplied by the fraction of the user-item matrix entries that are owned by the vendor.

dataset	D=3		D=5		D=7		D=9	
ML100K	0.06	86.6%	0.048	79.1%	0.036	72.9%	0.03	67.7%
ML1M	0.48	67.2%	0.3	54.7%	0.24	46.4%	0.24	39.6%
ML10M	179	96.9%	178	96.9%	177	96.9%	177	96.9%
ML20M	3270	99.1%	3257	99.1%	3253	99.1%	3252	99.1%
WN0	309	97.7%	308	97.7%	308	97.7%	3071	97.7%
WN1	766	98.1%	765	98.1%	765	98.1%	765	98.1%
WN2	1537	97.7%	1536	97.7%	1536	97.7%	1535	97.7%
WN3	3110	97.7%	3108	97.7%	3108	97.7%	3108	97.7%
WM0	309	97.7%	308	97.7%	308	97.7%	307	97.7%
WM1	1906	99.2%	1899	99.2%	1896	99.2%	1894	99.2%
WM2	7602	99.5%	7575	99.5%	7564	99.5%	7558	99.5%
WM3	30361	99.7%	30249	99.7%	30197	99.7%	30169	99.7%

Table 3. Runtimes (seconds) for a mediator in the offline phase

inner products in Line 1 of Protocol 2. That computation is carried out even in a non-private implementation of that CF method. Hence, the cost of privacy that our protocols entail is quite insignificant.

We can see from Table 3 that the runtime depends linearly on N and quadratically on M. Indeed, the protocol needs to compute $M(M-1)/2$ inner products of N-dimensional vectors, and as most of the runtime is spent on those computations, the overall runtime grows like $O(NM^2)$.

Table 3 does not show the runtime for the mediator in the single-mediator offline protocol of [38]; that is because in that protocol, all of the computations in the offline phase are done by the vendors. However, when considering the runtimes for both the vendors and mediators, as shown in Tables 2 and 3, it is evident that the improvement offered by distributing the mediation and, consequently, of replacing expensive homomorphic encryptions with lightweight secret sharing computations, is

overwhelming. Another advantage of our protocols is that the bulk of the computational overhead is transferred from the vendors to the mediators.

(We note that for the 20M dataset, [38] state that a direct implementation of their Protocol 2 would take several days. The time that they report in Table 2 there, of 48 minutes, is for a relaxed version of their Protocol 2 that re-uses encryptions of zero. Such a version is less secure than their original Protocol 2.)

Here we see that the number of mediators, D, has almost no effect on the runtime, and, in fact, as D increases the runtime for each mediator slightly decreases. We proceed to explain this phenomenon. The main computational effort is in the invocations of Protocol 2, which is called from Lines 11-13 in Protocol 1. The most time consuming computation here is the inner product that each mediator performs in Line 1 of Protocol 2; that computation is indifferent to D. However, the subsequent computation in Line 2 of Protocol 2 depends on D in two ways. On one hand, the time for reconstruction increases with D, since larger values of D require interpolation of higher-degree polynomials. On the other hand, the computational burden in Line 2 can be split between the mediators. Namely, the mediators can decide upfront that each one of them will take care of performing the computation in Line 2 of Protocol 2 only for $M(M-1)/(2D)$ pairs of items. A similar dual effect of D on the resulting runtimes exists also in the offline computations that are described in Sections 4.2.2 and 4.3. As it turns out, the latter effect (by which larger values of D help to reduce the average runtime per mediator by means of splitting the computational burden) is slightly more noticeable than the former effect (by which larger values of D require more involved computations).

5.2.3 Scenarios with competing vendors. To conclude our discussion of runtimes in the offline phase, we consider distribution scenarios in which different vendors own overlapping sub-matrices in the user-item matrix. Define $\alpha := \frac{\sum_{k\in[K]} N_k M_k}{NM}$ to be the *competition factor* of the distributed scenario; it equals the average number of competing vendors for each pair of a user and an item. In all distributed scenarios that were considered in prior art (see Section 3.2), $\alpha = 1$. The runtimes for the average mediator that were reported in Table 3 refer to the case $\alpha = 1$. If $\alpha > 1$, those runtimes increase very slightly. Table 4 reports the increase in runtime when $\alpha > 1$ (with respect to the runtimes when $\alpha = 1$, as reported in Table 3).

5.2.4 Runtimes of performing efficient updates. In Section 4.4 we described a way of executing the offline computations efficiently in wake of updates in the user-item rating matrix. Here we demonstrate the advantages of that approach.

Figures 1 and 2 show the offline runtimes for a vendor and for a mediator, on the two largest datasets (WN3 and WM3) as a function of the percentage of entries in the global user-item rating matrix R that were updated. In both figures we took the number of mediators to be $D = 9$, and we assumed that there is a single vendor that holds all of R (like we did in Section 5.2.1); as the runtimes for the vendors depend linearly on the size of the sub-matrix that they hold, the derivation of runtimes for vendors in a genuine distributed scenario from the runtimes shown in Figure 1 is straightforward. In Figure 2 we have also showed the standard deviation as we ran the experiment 50 times, each with different random entries to be updated. The percentage shown on the horizontal axis represents the number of entries for which the vendors send updates to the mediators at the end of an update period. Note that this percentage relates not only to entries that actually changed (A_k) but to the superset of entries for which updates are sent ($\overline{A}_k$). We chose to show runtimes for several percentage values, ranging from 1% to 100%. The runtimes shown for percentage of 100%, that is represented in both figures as a bold orange line, coincide with the runtimes that we reported earlier in Tables 2 and 3, since it reflects a full implementation of Protocol 1. The runtimes

dataset	α=2	α=3	α=4	α=5
ML100K	0.01	0.02	0.03	0.04
ML1M	0.147	0.294	0.441	0.588
ML10M	5.043	10.086	15.129	20.172
ML20M	24.445	48.89	73.335	97.78
WM0	6.6	13.2	19.8	26.4
WM1	13.2	26.4	39.6	52.8
WM2	33	66	99	132
WM3	66	132	198	264
WM4	132	264	396	528
WN0	6.6	13.2	19.8	26.4
WN1	13.2	26.4	39.6	52.8
WN2	33	66	99	132
WN3	66	132	198	264
WN4	132	264	396	528

Table 4. Overhead runtime (seconds) for a mediator in the offline phase for different values of α

shown for percentage of 1% illustrate the benefits of our suggested approach, as such a percentage seems to be a realistic upper bound on the fraction of entries that would need to be updated in typical update periods. Indeed, user-item rating matrices are typically very sparse (for example, the largest real dataset in our experiments, MovieLens 20M, has only 0.54% non-zero entries). Assume that during an update period 0.05% of the entries had changed (namely, about one tenth of the overall number of nonzero entries), and that the vendors used a superset $\overline{A}_k$ that is 20 times larger than A_k (what implies that the mediators could infer for each updated entry that it actually had changed in probability of 1/20). Then that would result in updating 1% of the matrix entries.

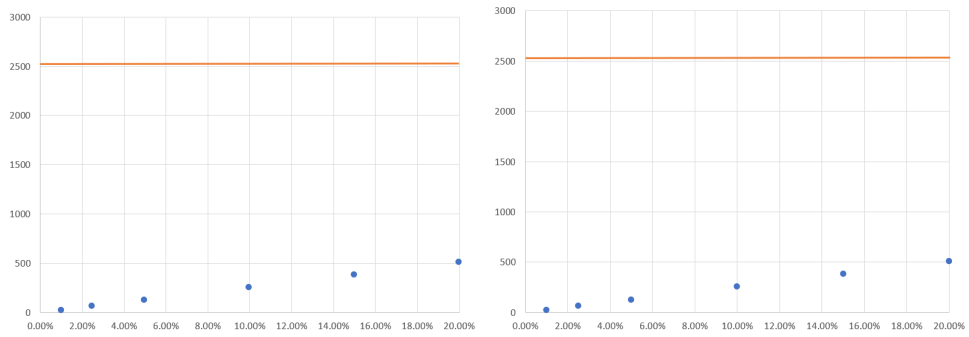

Fig. 1. Runtimes in seconds for a single vendor performing partial updates, as a function of the percentage of user-item matrix entries that were updated, on WN3 (top) and WM3 (bottom).

5.3 The online phase

The runtime for predicting ratings, for all testing configurations, was under 1 msec.[5] Computing the most recommended items (Protocol 3) is more time-consuming, as it requires computing M_k

[5]In all of our experiments in the online phase, we used $q = 80$ (see Section 3.1), as in [38].

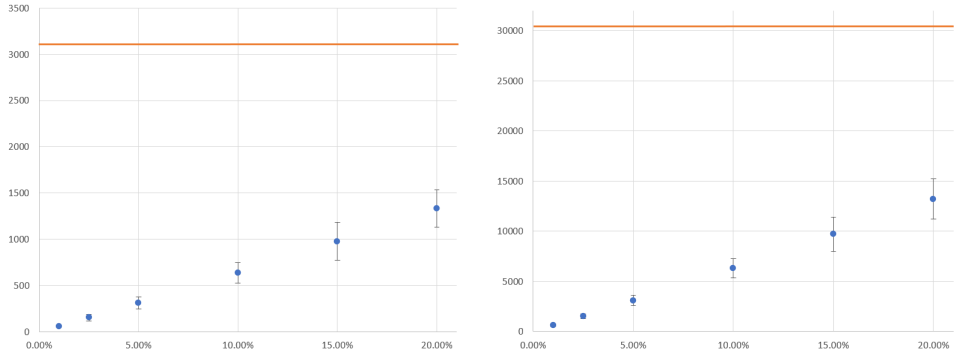

Fig. 2. Runtimes in seconds for each mediator performing partial updates, as a function of the percentage of user-item matrix entries that were updated, on WN3 (top) and WM3 (bottom).

inner products of M-dimensional vectors. Table 5 shows the runtimes of that computation for each of the datasets, under the assumption that $M_k = M$, for both our protocols and the single-mediator protocols of [38]. When $M_k < M$, all shown runtimes should be multiplied by M_k/M. (We do not consider here the WN0-WN3 datasets, since the runtimes in the online phase are not affected by N.) We note that also here, the advantage of distributing the mediation is manifested by a dramatic improvement in response times to recommendation queries.

dataset	Our protocol	SMP_{512} (V)		SMP_{512} (H)	
ML100K	<0.001	12.850	0.938	12.850	0.938
ML1M	<0.001	28.313	2.068	28.313	2.068
ML10M	0.044	81.572	5.959	81.572	5.959
ML20M	0.310	204.324	14.928	204.324	14.928
WM0	0.026	76.4	5.582	76.4	5.582
WM1	0.171	191	13.955	191	13.955
WM2	0.721	382	27.91	382	27.91
WM3	3.020	764	55.82	764	55.82

dataset	SMP_{1024} (V)		SMP_{1024} (H)	
ML100K	73.604	7.018	73.604	7.018
ML1M	162.174	15.465	162.174	15.465
ML10M	467.225	44.555	467.225	44.555
ML20M	1170.317	111.602	1170.317	111.602
WM0	437.6	41.73	437.6	41.73
WM1	1094	104.325	1094	104.325
WM2	2188	208.65	2188	208.65
WM3	4376	417.3	4376	417.3

Table 5. Runtimes (seconds) for computing the most recommended items. Top table: our distributed mediated protocol (left) and the Single-Mediator protocol of [38] with 512-bit encryptions, in the vertical (V) and horizontal (H) distribution scenarios, for the mediator (left value within each column) and for the vendor (right value). Bottom table: runtimes for the Single-Mediator protocol of [38] with 1024-bit encryptions. All runtimes in the tables need to be multiplied by M_k/M – the fraction of items offered by the corresponding vendor.

6 CONCLUSIONS

We presented herein secure multi-party protocols for performing item-based PPCF over distributed datasets for a general distribution scenario. Our protocols utilize mediation, in similarity to the protocols of [38], who showed the significant advantages of mediation in the context of PPCF. While their protocols used a single mediator, our protocols assume several independent mediators. That assumption enabled us to design protocols that are based on lightweight secret sharing computations rather than costly homomorphic encryption. Distributed mediation maintains all of the advantages of mediation that were identified in [38], and more: the distributed-mediator protocols offer stronger security and overwhelmingly shorter runtimes, in comparison to the single-mediator protocols. In addition, we considered a general distribution scenario that encompasses all previously considered distribution scenarios and extends them to a more realistic scenario, in which users have a choice between several competing vendors.

Some comments are in order:

(1) In Section 3.2 we stated our assumption that $I_k \subseteq [N]$ (the set of indices of users that V_k serves) and $J_k \subseteq [M]$ (the set of indices of the items that V_k offers) are publicly known. Specifically, for our purposes, it is required only that the mediators $\mathbf{T}$ know those sets, but not the other vendors. While such an assumption is very natural (since vendors typically do not wish to hide the set of items that they offer, nor the clientele which they serve), it is possible that a vendor of sensitive items would like to obfuscate those subsets. To protect the clientele subset I_k, such a vendor may publicize a set of users, $\hat{I}_k$, that extends the actual clientele that he serves, i.e., $I_k \subset \hat{I}_k \subseteq [N]$. Clearly, larger subsets $\hat{I}_k$ would offer greater privacy to the clientele of V_k, while setting $\hat{I}_k = [N]$ would provide the perfect privacy in that regard. However, such an artificial augmentation of the clientele set would incur an increased cost for V_k. Specifically, V_k's runtime in executing Lines 2-5 of Protocol 1 would increase by a factor of $|\hat{I}_k|/|I_k|$. (Similarly if V_k wishes to obfuscate J_k.) A vendor who wishes to protect one or both of those subsets may select the level of privacy desired according to the expected increase in runtime in the offline phase. It should be noted that such an action has a very modest effect on the mediators' runtime, since they would need to update more entries in their matrices (see Lines 8-9 in Protocol 1). However, such an action has no effect on the costlier Phase 3 of Protocol 1 (Lines 10-18) nor on the computations that are done in the online phase to reply to vendors' queries.

(2) In Section 4.1.2 we noted that our offline protocol is not perfectly secure, as it reveals also the intermediate values z_1, z_2, z_3. A more secure protocol would avoid the computation of those intermediate values and, instead, would use the shares that the mediators hold in each of the three inner products in the numerator and denominator in order to directly compute the desired fraction $S(\ell, m)^2 = z_1^2/(z_2 z_3)$, without exposing z_1, z_2, z_3. Such a computation can be carried out as follows. The operation of long division is just a sequence of multiplications, comparisons, and subtractions. If x and y are two integers, and the mediators have shares in them, it is well known how to compute shares in the product $x \cdot y$ [8, 10], in the bit that indicates whether $x < y$ [29], and in $x - y$ (straightforward). Hence, instead of recovering the three intermediate values z_1, z_2, z_3, the mediators can compute $S(\ell, m)^2 = z_1^2/(z_2 z_3)$ as follows: they will first compute shares in the numerator z_1^2 and in the denominator $z_2 z_3$ by performing two multiplications (along the lines of [8, 10]), and subsequently they will perform the long division procedure in order to recover shares in the desired value $S(\ell, m)^2$. From those shares they can then recover $S(\ell, m)$. However, such a computation is expected to have a severe toll on runtime. Given the benign nature of the information that is revealed through z_1, z_2, z_3, we believe that our current protocol is much more fitting for practical deployments, and we thus leave the design and evaluation of the more secure version which we outlined above to future research.

In summary, distributed mediation offers many attractive advantages in the context of PPCF: scalability; enhanced privacy; freeing the vendors from the need to communicate with each other, or the need to be online constantly; and enabling an economically-realistic collaboration model between the vendors. Hence, we intend to examine the applicability of distributed mediation to other CF algorithms, such as matrix factorization-based algorithms, e.g. [34], and compare their performance to that of existing privacy-preserving implementations of such algorithms, e.g. [27, 28].

REFERENCES

[1] Waseem Ahmad and Ashfaq A. Khokhar. 2007. An Architecture for Privacy Preserving Collaborative Filtering on Web Portals. In *The Third International Symposium on Information Assurance and Security, IAS*. 273–278.

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

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

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

[5] Anirban Basu, Jaideep Vaidya, Hiroaki Kikuchi, and Theo Dimitrakos. 2011. Privacy-preserving Collaborative Filtering for the Cloud. In *CloudCom*. 223–230.

[6] Anirban Basu, Jaideep Vaidya, Hiroaki Kikuchi, and Theo Dimitrakos. 2013. Privacy-Preserving Collaborative Filtering on the Cloud and Practical Implementation Experiences. In *IEEE CLOUD*. 406–413.

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

[8] Koji Chida, Daniel Genkin, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Yehuda Lindell, and Ariel Nof. 2018. Fast Large-Scale Honest-Majority MPC for Malicious Adversaries. In *CRYPTO*. 34–64.

[9] Richard Chow, Manas A. Pathak, and Cong Wang. 2012. A Practical System for Privacy-Preserving Collaborative Filtering. In *ICDM Workshops*. 547–554.

[10] Ivan Damgård and Jesper Buus Nielsen. 2007. Scalable and Unconditionally Secure Multiparty Computation. In *CRYPTO*. 572–590.

[11] 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.

[12] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. 2004. Efficient Private Matching and Set Intersection. In *EUROCRYPT*. 1–19.

[13] 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.

[14] Oded Goldreich, Silvio Micali, and Avi Wigderson. 2019. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography*. 307–328.

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

[16] 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.

[17] 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.

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

[19] 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.

[20] Hiroaki Kikuchi, Hiroyasu Kizawa, and Minako Tada. 2009. Privacy-Preserving Collaborative Filtering Schemes. In *ARES*. 911–916.

[21] Hiroaki Kikuchi and Anna Mochizuki. 2012. Privacy-Preserving Collaborative Filtering Using Randomized Response. In *IMIS*. 671–676.

[22] Donald E. Knuth. 1969. *The Art of Computer Programming, Volume II: Seminumerical Algorithms*. Addison-Wesley.

[23] 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.

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

[25] Yehuda Lindell and Ariel Nof. 2017. A Framework for Constructing Fast MPC over Arithmetic Circuits with Malicious Adversaries and an Honest-Majority. In *CCS*. 259–276.

[26] Burak Memis and Ibrahim Yakut. 2014. Privacy-Preserving Two-Party Collaborative Filtering on Overlapped Ratings. *KSII Trans. Internet Inf. Syst.* 8 (2014), 2948–2966.
[27] 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.
[28] Valeria Nikolaenko, Stratis Ioannidis, Udi Weinsberg, Marc Joye, Nina Taft, and Dan Boneh. 2013. Privacy Preserving Matrix Factorization. In *CCS*. 801–812.
[29] Takashi Nishide and Kazuo Ohta. 2007. Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol. In *PKC*. 343–360.
[30] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Eurocrypt*. 223–238.
[31] François Panneton, Pierre L'Ecuyer, and Makoto Matsumoto. 2006. Improved long-period generators based on linear recurrences modulo 2. *ACM Trans. Math. Softw.* 32 (2006), 1–16.
[32] Huseyin Polat and Wenliang Du. 2003. Privacy-Preserving Collaborative Filtering Using Randomized Perturbation Techniques. In *ICDM*. 625–628.
[33] Tal Rabin and Michael Ben-Or. 1989. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract). In *STOC*. 73–85.
[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.
[35] Assaf Schuster, Ran Wolff, and Bobi Gilburd. 2004. Privacy Preserving Association Rule Mining in Large Scale Distributed Systems. In *CCGRID*. 411–418.
[36] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11 (1979), 612–613.
[37] Erez Shmueli and Tamir Tassa. 2017. Secure Multi-Party Protocols for Item-Based Collaborative Filtering. In *RecSys*. 89–97.
[38] Erez Shmueli and Tamir Tassa. 2020. Mediated Secure Multi-Party Protocols for Collaborative Filtering. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 2 (2020), 1–25.
[39] Reza Shokri, Pedram Pedarsani, George Theodorakopoulos, and Jean-Pierre Hubaux. 2009. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. In *RecSys*. 157–164.
[40] Daiji Tanaka, Toshiya Oda, Katsuhiro Honda, and Akira Notsu. 2014. Privacy preserving fuzzy co-clustering with distributed cooccurrence matrices. In *SCIS&ISIS*. 700–705.
[41] Tamir Tassa. 2014. Secure Mining of Association Rules in Horizontally Distributed Databases. *IEEE Transactions on Knowledge and Data Engineering* 26 (2014), 970–983.
[42] 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.
[43] Tamir Tassa, Tal Grinshpoun, and Avishay Yanai. 2019. A Privacy Preserving Collusion Secure DCOP Algorithm. In *IJCAI*.
[44] 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.
[45] Tamir Tassa and Ehud Gudes. 2012. Secure Distributed Computation of Anonymized Views of Shared Databases. *Transactions on Database Systems* 37, Article 11 (2012).
[46] Tamir Tassa, Roie Zivan, and Tal Grinshpoun. 2015. Max-Sum Goes Private. In *IJCAI*. 425–431.
[47] Tamir Tassa, Roie Zivan, and Tal Grinshpoun. 2016. Preserving Privacy in Region Optimal DCOP Algorithms. In *IJCAI*. 496–502.
[48] Jaideep Vaidya and Chris Clifton. 2002. Privacy preserving association rule mining in vertically partitioned data. In *KDD*. 639–644.
[49] Udi Weinsberg, Smriti Bhagat, Stratis Ioannidis, and Nina Taft. 2012. BlurMe: inferring and obfuscating user gender based on ratings. In *RecSys*. 195–202.
[50] Ibrahim Yakut and Huseyin Polat. 2012. Arbitrarily Distributed Data Based Recommendations with Privacy. *Data & Knowledge Engineering* 72 (2012), 239–256.
[51] Andrew C. Yao. 1982. Protocols for Secure Computation. In *FOCS*. 160–164.
[52] Justin Zhijun Zhan, Stan Matwin, and LiWu Chang. 2005. Privacy Preserving Collaborative Association Rule Mining. In *Data and Applications Security*. 153–165.
[53] Sheng Zhong, Zhiqiang Yang, and Rebecca N. Wright. 2005. Privacy Enhancing k-Anonymization of Customer Data. In *PODS* (Baltimore, Maryland). 139–147.

A MULTIPLE RATINGS

So far we assumed that given any user u_n and item b_m, u_n had rated b_m at most once. Let us now assume that the user u_n (say, Alice) had purchased and rated b_m (say, the book "The Hobbit") through one vendor, V_{k_1}, and, as she liked it so much, she decided to purchase it also for her beloved niece. But this time she purchased it through another vendor, V_{k_2}, who offered it for a reduced price, and she submitted a rating for that item also through that vendor. Here we show that even in that scenario, the cosine similarity scores that Protocol 1 issues are meaningful.

Assume that the two ratings that u_n had submitted for b_m through V_{k_1} and V_{k_2} are r_1 and r_2, respectively. Then the n'th entry in the vector $\mathbf{c}_m$, in which the mediators hold shares, will be $r_1 + r_2$ (because V_{k_1} shared the value r_1 in that entry, V_{k_2} shared the value r_2, all other vendors shared a zero value in that entry, and the mediators add up the shares, see Lines 3 and 8 in the protocol). Similarly, the n'th entries in $\mathbf{c}_m^2$ and $\xi(\mathbf{c}_m)$ will be $r_1^2 + r_2^2$ and 2, respectively. We proceed to examine the effect that such double ratings have on the cosine similarity score that Protocol 1 computes for b_m against b_ℓ, for any $\ell \neq m$.

To that end, let us define the following vectors of dimensions $N + 1$:

$$\mathbf{c}_m^+ = (\mathbf{c}_m(1), \ldots, \mathbf{c}_m(n-1), r_1, r_2, \mathbf{c}_m(n+1), \ldots, \mathbf{c}_m(N))^T$$
$$\mathbf{c}_\ell^+ = (\mathbf{c}_\ell(1), \ldots, \mathbf{c}_\ell(n-1), \mathbf{c}_\ell(n), \mathbf{c}_\ell(n), \mathbf{c}_\ell(n+1), \ldots, \mathbf{c}_\ell(N))^T$$

Those would be the two column vectors for items b_m and b_ℓ over a population of $N + 1$ users which coincides with the original population of N users, except that Alice is replaced by two "reflections": Alice_1 and Alice_2. Both Alices rated all items the same as the real Alice, except for item b_m that Alice_1 rated r_1 and Alice_2 rated r_2. It is easy to see that the cosine similarity score that Protocol 1 computes for b_m and b_ℓ would be the same as that which would have been computed from $\mathbf{c}_m^+$ and $\mathbf{c}_\ell^+$ over the population of $N + 1$ users that includes the two reflections of Alice.

Namely, in such (probably rare) cases in which the same user purchases and rates the same item twice, the cosine similarity score regards those two actions as if they were carried out by two different users. It is easy to see that this interpretation extends also to cases in which there are several multiple ratings (namely, several pairs of a user u and an item b, in which u rated b through more than one vendor).

As a concluding note we observe that there is also another course of action for handling such cases. It is possible to identify upfront such multiple ratings and then retain just one of the ratings, say the latest one. However, such course of action would require every pair of vendors to communicate with each other; it would entail the invocation of costly Private Set Intersection [12] and secure comparison [51] sub-protocols; and it would disclose to the vendors sensitive information on purchases of users through other vendors. In addition, such treatment of multiple ratings completely ignores the fact that the user had liked the item so much that she or he had purchased and rated it more than once. Hence, Protocol 1 as is, without any modifications, offers the most suitable treatment of such multiple ratings, in terms of simplicity and efficiency, as well as in terms of utility for the purposes of CF.

Received XXX 2018