# Secure Multi-Party Protocols for Item-Based Collaborative Filtering

Erez Shmueli
Tel Aviv University
Tel Aviv, Israel
shmueli@tau.ac.il

Tamir Tassa
The Open University
Ra'anana, Israel
tamirta@openu.ac.il

## ABSTRACT

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

## CCS CONCEPTS

**•Human-centered computing → Collaborative filtering; •Security and privacy → Privacy-preserving protocols;**

## KEYWORDS

Item-based collaborative filtering, Distributed computing, Privacy

**ACM Reference format:**
Erez Shmueli and Tamir Tassa. 2017. Secure Multi-Party Protocols for Item-Based Collaborative Filtering. In *Proceedings of ACM Recommender Systems conference, Como, Italy, August 2017 (RecSys'17),* 9 pages.
DOI: 10.475/123_4

## 1 INTRODUCTION

The explosion of information that is available to users over the World Wide Web was the main driving force in the emergence of recommendation systems that aim at helping users find their needles in the haystack [23]. One of the main techniques on which recommendation systems are based is collaborative filtering (CF) [8]. CF predicts the preferences of users based on the preferences of the community (rather than on the users' characteristics). Specifically, CF methods use large databases that store information regarding rating or purchasing history of users in the community. These data are modeled as a matrix over the set of users (rows) and items (columns), where the entry for a given user $u$ and a given item $b$ is either binary (indicating whether user $u$ purchased item $b$ or not), or a rating that $u$ gave to $b$.

CF is broadly classified into memory-based and model-based approaches. The memory-based approach is either user-based, in the sense that recommendations for a given user $u$ are derived from the preferences of users that are similar to $u$, or item-based, i.e., $u$ is offered items which are similar to those that he purchased or liked in the past. The similarity between users or items is defined by some metric between the corresponding rows or columns, respectively, in the user-item matrix. The item-based approach is more suitable for deployment in e-commerce sites, due to its better scalability, and, indeed, it is one of the most widely deployed CF techniques [6].

Since memory-based approaches utilize the entire matrix of past ratings or purchases, some of them do not scale well. In such cases, model-based approaches may be more applicable. In model-based approaches, the original user-item matrix is used to train a compact model, which is then used for prediction. The model is developed by artificial intelligence techniques (e.g., Bayesian classification) or by linear algebraic techniques (e.g., SVD). While item-based methods have comparable performance to that of model-based methods when predicted ratings are the required output, the former methods tend to show better performance when the required output is the top $h$ items for a given user [6]. Furthermore, item-based recommendations are more easily interpretable to the user.

Instead of basing their recommendations to their clients solely on their own databases, vendors may significantly improve the quality of their recommendations by sharing their user-item preferential data [3]. However, such sharing poses significant privacy and security challenges. Indeed, commercial organizations may be reluctant to share their proprietary data about past users' purchases or ratings as it may serve competing entities. In addition, the users' privacy might be hindered if personal information about their past activities which they provided to one commercial entity would be handed over by that entity to other entities.

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

In this paper we propose protocols of Secure Multiparty Computation (SMC) for PPCF. We focus on the vertical distribution case, where each of the collaborating vendors (denoted $V_1, \ldots, V_K$) offers a different subset of items to the same underlying population of users. The discussion of the horizontal distribution setting, in which $V_1, \ldots, V_K$ offer the same set of items but each vendor serves a distinct subset of users, is deferred to the full version of this paper due to lack of space. We design privacy-preserving protocols for computing predicted ratings and rankings.

Most previous PPCF solutions are protocols that are executed solely by the vendors. Our protocols, on the other hand, involve a mediator, $T$. Having such a mediator reduces the need of the vendors to communicate with each other only to the offline (and infrequent) phase in which a model of similarity between items is built; in the online phase, where vendors need to issue predicted ratings and rankings to their clients, each vendor communicates directly vis-à-vis $T$ and does not need to communicate with other vendors. Such a mediated setting is advantageous over the non-mediated setting since it entails reduced communication costs. More importantly, each vendor can issue recommendations to its clients without "bothering" all other vendors each time or be dependent on their availability and willingness to collaborate.

The paper is organized as follows. We begin with a review of related work (Section 2). We then provide the necessary background on item-based CF and cryptographic methods (Section 3). Our PPCF protocols are described in Section 4. We present our experimental evaluation in Section 5, and conclude in Section 6.

## 2 RELATED WORK

Privacy-preserving collaborative filtering (PPCF) enables the practice of CF without leaking private information. We proceed to overview several recent PPCF works. Interested readers may refer to [4, 7] for a more comprehensive survey.

Polat and Du [19] discuss how to provide predictions for single items in case of a vertical distribution scenario with two vendors. In another study of theirs [20], they show how to offer top recommendations in both the horizontal and vertical distribution scenarios without deeply violating the vendors' privacy. They concentrate on the case of two vendors, and the case of binary user-item data. Jeckmans et al. [11] considered a case of horizontal distribution between two vendors, $V_1$ and $V_2$. They considered an asymmetric setting in which $V_2$ collaborates with $V_1$ so that $V_1$ can offer to its clients better recommendations, while $V_2$ does not benefit at all from such a collaboration. All of the above works are based on user-based CF, i.e., the recommendations to $u$ are derived from the items that users similar to $u$ liked. The studies that we describe next follow (like us) the item-based approach.

Basu et al. [2] propose algorithms for a privacy-preserving execution of an item-based CF scheme that is based on the Slope One predictor [15]. The work which is closest to ours is that of Yakut and Polat [33]. To the best of our knowledge, it is the only other work that deals with item-based CF which is based on the cosine-similarity score. They concentrated on the case of two vendors and considered the arbitrary distribution scenario. A main vehicle by which they obtain privacy is by introducing fake ratings into the distributed matrix. Hence, as opposed to our algorithms that issue exactly the same outputs as their non-privacy preserving counterparts, the outputs of the algorithms in [33] are inaccurate, and in order to enhance the privacy, greater levels of noise must be introduced, what implies a further reduction of accuracy.

All of the above studies are non-mediated. Hence, whenever a vendor wishes to compute predicted ratings or rankings, all other vendors need to be available and cooperating. Also, except [2], they concentrate on the case of $K = 2$ vendors; a natural extension of those schemes to a general $K$ (to enable better recommendations) entails high communication costs. Such challenges do not exist in the mediated setting, where any single vendor communicates (in the online phase) only with the mediator.

## 3 PRELIMINARIES

### 3.1 Item-based collaborative filtering

Here we provide a brief introduction to item-based CF [22]. A more comprehensive discussion is given in [6].

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

The similarity model is an $M \times M$ symmetric matrix $S$ where $S(\ell, m)$ is the similarity score between items $b_\ell$ and $b_m$, $\ell, m \in [M] := \{1, 2, \ldots, M\}$. Let $\mathbf{c}_m = (R(n, m) : n \in [N])$ denote hereinafter the $m$th column in the user-item rating matrix $R$, $m \in [M]$. Then the similarity scores are defined in Definition 3.1.[1]

*Definition 3.1.* Assume that $\ell, m \in [M]$ and set $\mathbf{c}_{\ell|m} := \mathbf{c}_\ell \cdot \xi(\mathbf{c}_m)$; namely, $\mathbf{c}_{\ell|m} := (\mathbf{c}_{\ell|m}(1), \ldots, \mathbf{c}_{\ell|m}(N))^t$ is the projection of the $\ell$th column of the user-item rating matrix $R$ on the subset of users that rated both items $b_\ell$ and $b_m$. Then the cosine similarity score is

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

The similarity scores are used to predict $u_n$'s rating of $b_m$ as follows. Let (a) $N_q(m)$ be the set of indices of the $q$ items with highest $S(\cdot, m)$ (for some preset, typically small, $q < M$); (b) $N_q^+(m) := \{\ell \in N_q(m) : S(\ell, m) > 0\}$; (c) $\mathbf{s}_m$ be the vector for which $\mathbf{s}_m(\ell) = S(\ell, m)$ if $\ell \in N_q^+(m)$ and $\mathbf{s}_m(\ell) = 0$ for $\ell \in [M] \setminus N_q^+(m)$ ; (d) $\overline{R(b_m)}$

[1]Notation agreements: (a) If $r$ is a nonnegative integer then $\xi(r) = 0$ if $r = 0$ and $\xi(r) = 1$ otherwise. (b) If $\mathbf{x}$ is a vector and $f$ is a function then $f(\mathbf{x})$ is the vector in which $f(\mathbf{x})(\cdot) = f(\mathbf{x}(\cdot))$. (c) If $\mathbf{x}$ and $\mathbf{y}$ are vectors then $\langle \mathbf{x}, \mathbf{y} \rangle$ is their inner product, while $\mathbf{x} \cdot \mathbf{y}$ is the vector in which $(\mathbf{x} \cdot \mathbf{y})(\cdot) = \mathbf{x}(\cdot) \cdot \mathbf{y}(\cdot)$.

be $b_m$'s average rating; (e) $\mathbf{r}_n$ be the $n$th row of the matrix $R$; and (f) $\bar{\mathbf{r}}_n$ be the vector of $u_n$'s adjusted ratings, i.e. $\bar{\mathbf{r}}_n(m) = (R(n,m) - \overline{R(b_m)}) \cdot \xi(R(n,m))$, $m \in [M]$. Then

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

If the denominator in Eq. (2) equals zero, $P(u_n, b_m)$ is set to $\overline{R(b_m)}$. (There exist several variants of the similarity score and prediction formula. We focus here on the variant suggested in [6], which performs best. The adjustment of our protocols to other variants is straightforward.)

Sometimes, instead of showing to $u_n$ his predicted rating on some item, the goal is to present to him the $h$ items which are most likely to appeal to him, without predicted ratings. To that end, one produces a *ranking* of all items that $u_n$ had not rated so far in order to extract from it the top $h$ items, for some $h \geq 1$. One approach is to base the ranking on the items' predicted ratings (Eq. (2)). However, a simpler ranking procedure produces better results (see [6, Section 2.3]). Let $I(n)$ be the subset of indices of items that $u_n$ already rated. Define for each $m \in [M] \setminus I(n)$ the score

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

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

## 3.2 Cryptographic building blocks

An encryption function $\mathcal{F}$ is called (additively) *homomorphic* if for every two plaintexts, $m_1$ and $m_2$, $\mathcal{F}(m_1 + m_2) = \mathcal{F}(m_1) \cdot \mathcal{F}(m_2)$. When the encryption function is randomized (in the sense that $\mathcal{F}(m)$ depends on $m$ as well as on a random string) then $\mathcal{F}$ is called *probabilistic*. The semantically secure Paillier cipher [18] is both homomorphic and probabilistic. In our protocols, $V_1, \ldots, V_K$ jointly generate the key pair in a homomorphic and probabilistic encryption function $\mathcal{F}$; they notify $T$ of the encryption key, but keep the decryption key private. (In order to jointly generate a random key pair, the vendors only need to generate a random bit string of the same length as the key; this can be done, say, if every vendor chooses its own random string and then they all engage in a secure summation protocol.)

Another building block of secure multi-party computation that we shall need is that of secure division. Tassa and Bonchi [26] considered a setting that involves three parties, $P_1$, $P_2$ and $H$. $P_i$, $i = 1, 2$, holds a private integer $a_i$, and the goal is to let $H$ recover the real quotient $a_1/a_2$ but not the values of $a_1$ and $a_2$. Towards that end, $P_1$ and $P_2$ jointly generate a random real number $X \sim Z$ where $Z$ is the distribution on $[1, \infty)$ with probability density function $f_Z(x) = x^{-2}$; then they jointly generate a random $g \sim U(0, X)$, and they send to $H$ the values $ga_i$, $i = 1, 2$, which $H$ proceeds to divide in order to recover $q = a_1/a_2$. The masking multiplier $g$ prevents $H$ from learning $a_i$, but it can still recover $q$. The selection of the probability distribution from which $g$ is drawn is made in order to minimize the information that $H$ can extract from $ga_i$ on $a_i$. Here we use this idea as follows: $T$ holds two encrypted values $\mathcal{F}(a_1)$ and $\mathcal{F}(a_2)$ and it wishes to allow a vendor $V_k$ to get the quotient $q = a_1/a_2$ so that no one reveals $a_1$ nor $a_2$. To that end it sends to $V_k$ the values $b_i := \mathcal{F}(a_i)^g$, $i = 1, 2$, for a random $g$ that was generated as described above. Since $\mathcal{F}$ is homomorphic then $b_i := \mathcal{F}(ga_i)$. $V_k$ decrypts the two received values and gets $\mathcal{F}^{-1}(b_i) = ga_i$, $i = 1, 2$, which it then proceeds to divide in order to get $q$.

## 4 PPCF PROTOCOLS

Here we deal with the vertical distribution scenario, where each vendor owns a different subset of $R$'s columns. In Sections 4.1 and 4.2 we describe the computations and protocols that are performed in the offline phase and involve all of the vendors $V_1, \ldots, V_K$, as well as the mediator $T$. Then, we describe the online phase in which a given vendor $V_k$ submits queries to $T$ towards computing the predicted rating of some user $u_n$ for an item $b_m$, $P(u_n, b_m)$, Eq. (2) (Section 4.3), or getting the top $h$ items for a given user $u_n$ (Section 4.4). The online phase is carried out solely by $V_k$ and $T$. Namely, the participation of all vendors is required only in the offline and less frequent phase.

**User ordering.** The vendors jointly decide on a random ordering of $U$, which is kept secret from $T$. Hereinafter, $u_n$ denotes the $n$th user in that ordering.

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

We note that in order to select those random orderings, all that is needed is to select a common random seed; such a common random seed can be generated as described in Section 3.2.

**Privacy.** The goal of each vendor $V_k$ is to protect its own proprietary data, being the sub-matrix $R_k := \{R(n,m) : n \in [N], m \in I_k\}$, from $V_j$, $j \neq k$, and $T$. Ideally, no such party should gain any information on $R_k$ from its view during the execution of our protocols, beyond what can be inferred from its own input and the output of the protocol. However, while such perfect security is achievable for any problem of SMC (by invoking generic solutions such as Yao's garbled circuit construction [34]), 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. association rule mining [13, 25, 31], anonymization [27, 28, 35], computation on distributed graphs [1] or constraint optimization [9, 10, 14, 29, 30]. In fact, also all PPCF studies that were reviewed in Section 2 offer protection to the private data, but that protection is not perfect. The protocols that we present here are also privacy-preserving in that sense. For each of our protocols we discuss its privacy-preservation and identify the excess information that they may leak, and explain why such leakage of information is benign. Our protocols rely solely on standard cryptographic building blocks, as described in Section 3.2; this offers a significant advantage as they can be readily implemented on top of standard libraries.

We make here an assumption which is common in PPCF literature: all parties are semi-honest and do not collude. Semi-honesty means that the parties follows the protocols' specifications but try to extract from their view information on the inputs of their peers.

(See, for instance, [12, 24, 35] for a discussion and justification of that assumption.)

## 4.1 Offline model construction

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

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

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

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

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

Secure computation of scalar products of private vectors is a fundamental problem in SMC, as it serves a basic building block for many other secure protocols (see e.g. [32] and the references therein). Protocol 1 can be executed with any SSP sub-protocol. We find the protocol of Du and Zhan [5] most fitting in our context, since their protocol is designed for the mediated setting which we consider; namely, two parties hold each of the input vectors and the scalar product goes to a mediator. In addition, because it relies on a mediator, it solves the problem with perfect security without resorting to expensive cryptographic means, thus implying low computational and communication costs.

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

**Privacy.** By using a secure SSP sub-protocol (such as the one in [5]), none of the vendors may infer any information on inputs of other vendors. As for $T$, the usage of random multipliers prevents it from learning the terms in the numerator and denominator of Eq. (1). $T$ does learn the similarity scores between items, but, owing to the random item ordering, it cannot link those scores to the relevant pair of items.

(Due to page limitation, we omit the computational and communication cost analysis of all our protocols.)

**A concluding remark.** The similarity scores $S(\ell, m)$ are real-valued and in the next phase they need to be used in computing predicted ratings and rankings in a privacy-preserving manner. In doing so, it is necessary to subject them to encryption. Since encryption functions are applied on discrete integer domains, $T$ translates all real values $s$ in the matrix $S$ into integer values by mappings of the form $s \mapsto \hat{s} := \lfloor Ls + 0.5 \rfloor$, where $L$ is a large integer.

## 4.2 In preparation to the online phase

The computation of predicted ratings depends on $\bar{\mathbf{r}}_n$ and $\xi(\mathbf{r}_n)$, see Eq. (2). The vectors $\bar{\mathbf{r}}_n$ are real-valued. We translate them into integer-valued vectors $\hat{\mathbf{r}}_n$, where $\hat{\mathbf{r}}_n(m) := \lfloor L\bar{\mathbf{r}}_n(m) + 0.5 \rfloor$, and $L$ is a large integer as described in the concluding remark of Section 4.1. Then, in the online phase, the predicted ratings will be computed as follows:

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

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

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

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

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

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

Note that practical deployments of our protocols should be enhanced with standard security mechanisms. In particular, each party should create its own pair of private and public keys, and get from a Certificate Authority a corresponding certificate. Then, each message must be signed by the sender and encrypted under the receiver's public key. Such standard security layers are essential when implementing *any* SMC protocol, and they come *on top* of the SMC protocol. They are essential in order to prevent eavesdropping, masquerading, and other well-known attacks on communication systems.

## 4.3 Computing predicted ratings

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

Lemma 4.1. *Let $\mathcal{F}$ be a homomorphic encryption function and let $\mathbf{x}$ and $\mathbf{y}$ be two $N$-dimensional integer vectors. Denote $(\mathcal{F}(\mathbf{x}))^{\mathbf{y}} := \prod_{n=1}^{N} \mathcal{F}(\mathbf{x}(n))^{\mathbf{y}(n)}$. Then $(\mathcal{F}(\mathbf{x}))^{\mathbf{y}} = \mathcal{F}(\langle \mathbf{x}, \mathbf{y} \rangle)$.*

Protocol 3 begins with $V_k$ submitting a query to $T$ (Step 1); the query includes an index $n \in [N]$ of a user $u_n \in U$ and an index $m \in I_k$ of an item $b_m$ that $V_k$ offers. Then, $T$ computes the set $N_q^+(m)$ (see Section 3.1); those are the indices $\{\ell_1, \ldots, \ell_t\}$ of the $t \le q$ items in $B \setminus \{b_m\}$ (not necessarily from among those offered by $V_k$) that have the highest and positive similarity scores against $b_m$ (Step 2). $T$ then sets $\mathbf{s}_m$ to be the $m$th column of the similarity matrix, where all entries not corresponding to the above $N_q^+(m)$-items are zeroed (Step 3). Those steps can be carried out just once for each item, so that the results can be used in future queries for the same item.

In Step 4 $T$ selects a random multiplier $g$ (as explained in Section 3.2) that will obfuscate from $V_k$ the values of the numerator and the denominator in the quotient in Eq. (4), but will enable $V_k$ to compute that quotient. Recall that, owing to Protocol 2, $T$ has the vectors $\mathcal{F}(\hat{\mathbf{r}}_n)$ and $\mathcal{F}(\xi(\mathbf{r}_n))$ for $u_n$. Then, relying on Lemma 4.1, $T$ computes in Step 5 the $\mathcal{F}$-encryption $x$ of the scalar product $x' := g \cdot \sum_{i=1}^{t} S(\ell_i, m)\hat{\mathbf{r}}_n(\ell_i)$, as well as the $\mathcal{F}$-encryption $y$ of the scalar product $y' := g \cdot \sum_{i=1}^{t} S(\ell_i, m)\xi(\mathbf{r}_n(\ell_i))$. It sends those values to $V_k$ who proceeds to decrypt them (Step 6) and then use them in Eq. (4) to get the predicted rating (Step 7). (Recall that the average rating for each item $b_m$, $m \in I_k$, can be computed by $V_k$ alone and was already computed in Protocol 2.)

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

## 4.4 Computing the most recommended items

Here we discuss the case where in the online phase, instead of showing to the user $u_n$ predicted ratings for items that he had not rated so far, $V_k$ presents to him a list of the $h$ items, yet unrated by $u_n$, which are most likely to appeal to him. In view of our discussion in Section 3.1, such a computation would be carried out by $V_k$ and

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

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

$T$ as follows: if $I(n)$ is the subset of indices of items that $u_n$ already rated, then $V_k$ will select the $h$ indices $m \in I_k \setminus I(n)$ for which $\hat{s}(m) = \sum_{\ell \in I(n) \cap N_q(m)} S(\ell, m)$, Eq. (3), are largest, where $S(\ell, m)$ are the scaled integral similarity scores that $T$ got as a result of Protocol 1.

Protocol 4 performs that computation in a privacy-preserving manner. It starts by $V_k$ submitting a query to $T$ (Step 1). In Steps 2-4, $T$ generates an $\mathcal{F}$-encryption of $\hat{s}(m)$ for all $m \in I_k$, multiplied by a random integer multiplier $g$, as described in Section 3.2. Indeed, by Eq. (3) and Lemma 4.1, and the fact that $\xi(\mathbf{r}_n)$ is a binary vector with $\xi(\mathbf{r}_n)(\ell) = 1$ if and only if $\ell \in I(n)$, it follows that $\mathbf{x}(m) = \mathcal{F}(g\hat{s}(m))$ for all $m \in I_k$.

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

Next, $T$ sends a random permutation of the vectors $\mathbf{x}$ and $\mathbf{y}$ to $V_k$, who decrypts them into $\mathbf{x}'$ and $\mathbf{y}'$, respectively (Steps 6-7). Hence, $\mathbf{x}'$ holds a permutation of the values $g\hat{s}(m)$, for all items $m \in I_k$, while $\mathbf{y}'$ holds a corresponding permutation of 0 and 1 values that identify those items which $u_n$ had already rated. Since $T$ had scrambled the latter ciphertexts (by multiplying each one of them with a fresh random encryption of 0), $V_k$ can only distinguish between rated and unrated items, but it cannot distinguish between items within either one of those two subsets. (If $T$ had not multiplied each $\mathcal{F}(\xi(\mathbf{r}_n)(m))$ with a fresh encryption of zero, then $V_k$ would have been able to reverse engineer the random permutation $\pi$ by comparing the entries of $\mathbf{y}$ with the encrypted entries of $\xi(\mathbf{r}_n)$ that it had sent to $T$ earlier in Step 4 in Protocol 2.) $V_k$ proceeds to find the indices of the $h$ yet unrated items with largest $g\hat{s}(m)$ (and hence also largest $\hat{s}(m)$) and sends them to $T$ (Step 8). $T$ responds by letting $V_k$ know the original indices of those items (Step 9). Those are the items to be presented to the user $u_n$ as the top-$h$ recommended items.

**Privacy.** The values of the scores $\hat{s}(m)$ are hidden from $V_k$ by the random multiplier $g$. However, using that mechanism alone

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

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

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

## 5 EXPERIMENTS

**Experimental setting.** All experiments were run on a 13-inch MacBook Pro with a 3.0GHz dual-core Intel Core i7 CPU and 16GB of RAM. The algorithms were implemented in Java as an extension to the open source LibRec library[2].

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

**Table 1: Dataset characteristics**

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

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

In the first two experiments, we compare the performance of our protocols for predicting ratings and rankings to the protocol of Yakut and Polat (YP) [33], which is the only other item-based PPCF protocol that is directly comparable to ours. Since the latter protocol is designed for the case $K = 2$, our next experiments are carried out on vertical splits of the user-item matrices into two random vertical sub-matrices.

One of the tools which YP utilizes in order to offer privacy is by adding fake ratings. Assume that $V_k$ has a sub-matrix of dimensions $N \times M_k$ that includes $numR_k$ actual ratings. Then $V_k$ replaces $p$% of the remaining $NM_k - numR_k$ entries with fake ratings, $k = 1, 2$. The value of the fake ratings can be set in several ways. One of the suggestions made in [33], which we adopt herein, is that each $V_k$ uses the global mean rating over all rated entries in its sub-matrix.

**Experiment 1.** We compared the accuracy of our Protocol 3 for predicting ratings to that of YP. We applied Protocol 3 for each of the testing entries and then computed the MAE over all testing entries (in both sites of $V_1$ and $V_2$). We did a similar test with the prediction protocol YP, with several values of $p$ (the percentage of fake ratings). Finally, we used a baseline algorithm that simply predicts for each user $u$ and item $b$ that $P(u, b) = \overline{R(b)}$, where $\overline{R(b)}$ is the average rating given to item $b$. Figure 1 shows the resulting errors as obtained on each of our datasets. (Recall that the shown values are averaged over ten random and independent runs.) As can be seen, the accuracy of YP with $p = 0$ coincides with that of our Protocol 3, but such a version of YP is non-private. Increasing $p$ to higher values that would provide better hiding of each vendor's data results in higher errors, that in the two MoviLens datasets even became higher than that of the naïve (and perfectly secure) item-mean predictor. To summarize, our protocols, which base their security on cryptographic means, rather than randomization, offer higher security than YP; in addition, the output of our protocols (as opposed to YP) fully coincides with the non-secure algorithm.

**Experiment 2.** Next, we compared the quality of ranking as offered by our Protocol 4, to the quality of rankings which are derived from computing predicted ratings, by either our Protocol 3 or by YP [33]. The comparison is made by the AUC (Area Under the Receiver Operating Curve) measure; AUC values range from 0.5 (worst) to 1 (best).

Let $I_{k,n}$ deote the set of items offered by $V_k$ which $u_n$ had rated, and their corresponding entries in $R$ were selected for training. We follow the usual practice of generating rankings over all items in the complement set, $I^c_{k,n} := I_k \setminus I_{k,n}$, and then comparing that ranking to the baseline ranking, in which all items in $I^c_{k,n}$ which $u_n$ had rated are considered positive and all other items in $I^c_{k,n}$ are considered negative. Specifically, for each $m \in I^c_{k,n}$ we computed

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

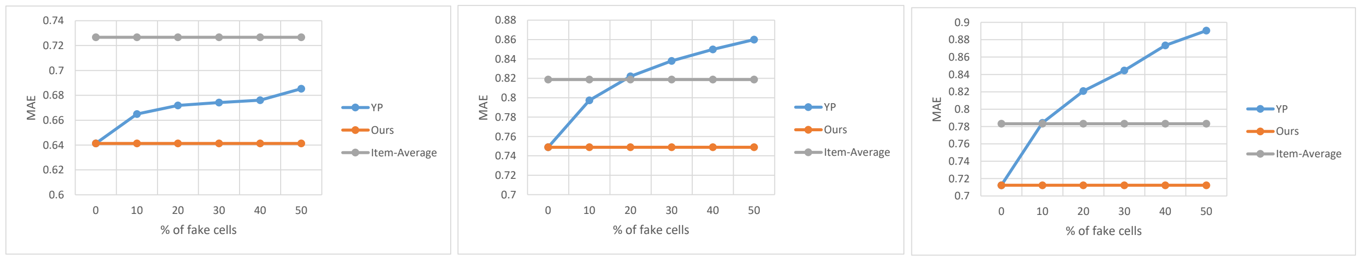

**Figure 1: Rating prediction: FilmTrust (left), MovieLens 100K (center) and MovieLens 1M (right)**

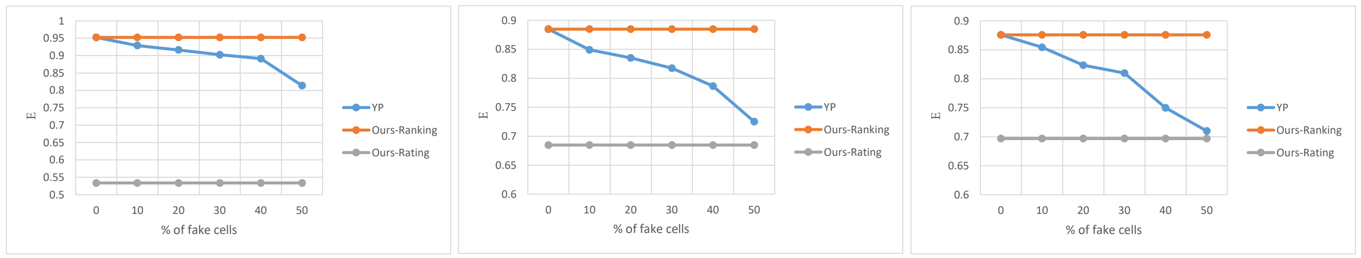

**Figure 2: Ranking prediction: FilmTrust (left), MovieLens 100K (center) and MovieLens 1M (right)**

three scores: its predicted rating by Protocol 3, its predicted rating by YP, and the score $\hat{s}(m)$, Eq. (3), by which Protocol 4 ranks. By thus we get three sequences of scores which induce three rankings over $I^c_{k,n}$. We compared each of those rankings to the above described baseline ranking. Letting $E_{k,n}$ denote the AUC value obtained for one of those three rankings, we finally compute the average value $E = \left(\sum_{k=1}^{2} \sum_{n=1}^{N} E_{k,n}\right)/2N$. Figure 2 shows the average of $E$, for each of the three scoring functions, over ten random runs. As can be seen, Protocol 4, which is designed for issuing rankings, issues significantly better rankings than rankings based on the predicted ratings of either Protocol 3 or YP.

**Experiment 3.** We conclude with runtime experimentation. We measured the cryptographic overhead on both the mediator $T$ and a typical vendor $V_k$ in the offline and online phases. For encryption we used the Paillier cryptosystem [18] implemented in Java[3]. Table 2 shows the runtime overheads in each of the three datasets, when they are distributed evenly and vertically among $K = 5$ vendors[4], in hours(h), minutes(m), seconds(s), and miliseconds(ms).

In the offline phase, we focus only on Protocol 2. We ignore Protocol 1 for two reasons. First, the actual runtime depends on the selection of the SSP sub-protocol. Second, if one chooses the SSP protocol of [5], which is most fitting in our setting, then it involves no cryptographic operations, and hence the privacy-induced overhead is negligible. Protocol 2 entails cryptographic overhead only on the vendors, as they need to encrypt their matrix entries. The overhead is considerable. In the three datasets apart from the largest one, MovieLens 20M, the offline computation time takes from 14.5 minutes up to 3.4 hours. However, as that phase is run only once in a period (say, once a week), and since the required computations can be easily parallelized, such overhead is reasonable.

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

In order to dramatically decrease the runtime, we created, for each vendor, a "dictionary" of $\sqrt{2NM_k}$ random and independent encryptions of zero. Then, whenever we had to create a random encryption of zero, we picked two elements from the dictionary and multiplied them (without picking the same pair twice). That way, we were able to generate the needed number of zero encryptions without repeating the same ciphertext twice. Since the cost of multiplication is much smaller than that of encryption, the resulting runtime was reduced to only 48 minutes.

For the online phase we used $q = 80$. (Recall that $q$ is the size of the item-neighborhoods which are used in predicting ratings

[3]www.csee.umbc.edu/~kunliu1/research/Paillier.html
[4]The times in the Rating column are independent of $K$. Those in the Offline and Ranking columns are inversely proportional to $K$, under the assumption of even distribution.

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

**Table 2: Cryptographic runtime overheads**

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

## 6 CONCLUSION

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

This study suggests several future research directions: (a) Enhancing our protocols so that they offer privacy even if some of the interacting parties do not act honestly, or collude. To the best of our knowledge, none of the existing PPCF studies had considered such settings. (b) Generalizing our protocols to deal with an arbitrary distribution setting (i.e, not a purely vertical or a purely horizontal split). (c) Examining the applicability of our techniques to other CF algorithms, such as matrix factorization-based algorithms and their extensions (e.g. [21]) and compare their performance to that of existing privacy-preserving matrix factorization algorithms, such as [16, 17].

## REFERENCES

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

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

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

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

[5] Wenliang Du and Justin Zhijun Zhan. 2002. A practical approach to solve Secure Multi-party Computation problems. In *Workshop on New Security Paradigms*. 127–135.

[6] 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.

[7] Arik Friedman, Bart P Knijnenburg, Kris Vanhecke, Luc Martens, and Shlomo Berkovsky. 2015. Privacy aspects of recommender systems. In *Recommender Systems Handbook*. 649–688.

[8] 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.

[9] Tal Grinshpoun and Tamir Tassa. 2014. A privacy-preserving algorithm for distributed constraint optimization. In *AAMAS*. 909–916.

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

[11] Arjan Jeckmans, Qiang Tang, and Pieter H. Hartel. 2012. Privacy-preserving collaborative filtering based on horizontally partitioned dataset. In *International Conference on Collaboration Technologies and Systems*. 439–446.

[12] Wei Jiang and Chris Clifton. 2006. A secure distributed framework for achieving $k$-anonymity. *The VLDB Journal* 15 (2006), 316–333.

[13] 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.

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

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

[16] 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.

[17] Valeria Nikolaenko, Stratis Ioannidis, Udi Weinsberg, Marc Joye, Nina Taft, and Dan Boneh. 2013. Privacy-preserving matrix factorization. In *CCS*. 801–812.

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

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

[20] Huseyin Polat and Wenliang Du. 2008. Privacy-preserving top-$N$ recommendation on distributed data. *Journal of the Association for Information Science and Technology* 59 (2008), 1093–1108.

[21] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.

[22] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.

[23] J. Ben Schafer, Joseph A. Konstan, and John Riedl. 1999. Recommender systems in e-commerce. In *EC*. 158–166.

[24] Assaf Schuster, Ran Wolff, and Bobi Gilburd. 2004. Privacy-preserving association rule mining in large-scale distributed systems. In *CCGRID*. 411–418.

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

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

[27] 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.

[28] Tamir Tassa and Ehud Gudes. 2012. Secure distributed computation of anonymized views of shared databases. *Transactions on Database Systems* 37, Article 11 (2012).

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

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

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

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

[33] Ibrahim Yakut and Huseyin Polat. 2012. Arbitrarily distributed data-based recommendations with privacy. *Data & Knowledge Engineering* 72 (2012), 239–256.

[34] Andrew C. Yao. 1982. Protocols for secure computation. In *FOCS*. 160–164.

[35] Sheng Zhong, Zhiqiang Yang, and Rebecca N. Wright. 2005. Privacy-enhancing $k$-anonymization of customer data. In *PODS*. 139–147.