Future Generation Computer Systems 55 (2016) 311–320

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An algorithm for efficient privacy-preserving item-based collaborative filtering

Dongsheng Li [a], Chao Chen [a], Qin Lv [b,*], Li Shang [a,b], Yingying Zhao [a], Tun Lu [c], Ning Gu [c,*]

[a] Tongji University, Shanghai 201804, PR China
[b] University of Colorado Boulder, Boulder, CO 80309, USA
[c] Fudan University, Shanghai 200433, PR China

HIGHLIGHTS

- We propose an efficient privacy-preserving item-based collaborative filtering method.
- We propose an unsynchronized protocol to achieve secure multi-party computation.
- We propose two incremental privacy-preserving item similarity computation methods.
- The privacy preservation property of the proposed method is formally proved.
- The proposed method is more efficient and accurate than two well-known methods.

ARTICLE INFO

Article history:
Received 20 November 2013
Received in revised form
10 September 2014
Accepted 10 November 2014
Available online 2 December 2014

Keywords:
Item-based
Collaborative filtering
Privacy
Efficiency

ABSTRACT

Collaborative filtering (CF) methods are widely adopted by existing recommender systems, which can analyze and predict user "ratings" or "preferences" of newly generated items based on user historical behaviors. However, privacy issue arises in this process as sensitive user private data are collected by the recommender server. Recently proposed privacy-preserving collaborative filtering (PPCF) methods, using computation-intensive cryptography techniques or data perturbation techniques are not appropriate in real online services. In this paper, an efficient privacy-preserving item-based collaborative filtering algorithm is proposed, which can protect user privacy during online recommendation process without compromising recommendation accuracy and efficiency. The proposed method is evaluated using the Netflix Prize dataset. Experimental results demonstrate that the proposed method outperforms a randomized perturbation based PPCF solution and a homomorphic encryption based PPCF solution by over 14X and 386X, respectively, in recommendation efficiency while achieving similar or even better recommendation accuracy.

1. Introduction

Recommender systems are becoming important due to the increasing "information overload" challenge on the Internet. Collaborative filtering (CF), as one of the most popular recommendation techniques, is adopted by many online service providers, such as Amazon [1], Youtube [2] and Google News [3]. CF methods work as follows: (1) the server first collects user historical behaviors and analyzes user/item correlations; and (2) recommendations are generated based on these user/item correlations. During this process, user specific, hence, sensitive information, such as item rating, demographical information, activity pattern, social relationships, etc., are collected by the recommender system, which arises privacy concerns. Recent studies have shown that user privacy could be exploited by service providers or malicious users to gain profits. Recommender system (service provider) could share user private data with other parties to make personalized advertisements [4], or even sell these information to other parties [5]. In some cases, user private data may be exposed via open APIs of service provider [6], or attacked by malicious users [7,8]. To eliminate these concerns, CF methods should not disclose user private data to the recommender system yet allowing to provide personalized recommendations to end users.

Existing works on privacy-preserving collaborative filtering (PPCF) mainly rely on cryptography [5,9,10] or data perturbation [11–13] to protect the recommender system from obtaining user

* Corresponding authors.
E-mail addresses: qin.lv@colorado.edu (Q. Lv), ninggu@fudan.edu.cn (N. Gu).

http://dx.doi.org/10.1016/j.future.2014.11.003
0167-739X/

data. Cryptography based PPCF methods generally adopt homomorphic encryption to encrypt user information. Then, recommendation scores could be computed on encrypted data, so that user privacy are protected. It is known that, complex encryptions and decryptions are computationally prohibitive for large-scale online services which face with serious scalability issues. Data perturbation based PPCF methods inject noise on user data to prevent recommender system from obtaining user privacy. However, this kind of methods decrease the accuracy of recommendations [11–13]. It is known that accuracy is the ultimate goal of recommender system, so that degradations in recommendation accuracy are not acceptable in real online services. Moreover, data perturbation technique is limited in privacy preservation. Recent works [14,15] have shown that the server could partially recover user privacy from perturbed user data using machine learning techniques. In summary, it calls for a collaborative filtering method which offers high efficiency and high accuracy and protects user privacy for large-scale online services.

In this paper, an efficient privacy-preserving item-based collaborative filtering algorithm is proposed, which can protect user privacy during recommendation process without compromising accuracy and efficiency. In the proposed method, item similarities are calculated by efficient secure multi-party computation (SMPC), which can achieve the same efficiency and accuracy as centralized item similarity computation. After similarity computation, users could locally calculate recommendation scores and obtain recommendations with privacy. The proposed method is evaluated on the Netflix Prize dataset, and experimental results demonstrate that the proposed method can achieve higher efficiency than two well-known PPCF solutions without compromising recommendation accuracy. The contributions of this work are summarized as follows:

1. An efficient privacy-preserving item-based collaborative filtering algorithm is proposed to protect user privacy during recommendation process without compromising recommendation accuracy and efficiency.
2. An unsynchronized secure multi-party computation protocol is proposed to achieve multi-party computation without requiring that users should be online simultaneously during computation.
3. Two similarity computation algorithms are proposed to efficiently measure item similarities without compromising user privacy. Meanwhile, the proposed methods could incrementally compute item similarities, so that the item similarity model could be updated incrementally in the proposed method.
4. The proposed method is evaluated on the Netflix Prize dataset. Experimental results demonstrate that the overall efficiency of the proposed method outperforms a randomized perturbation based PPCF method and a homomorphic encryption based PPCF method by over 14X and 386X, respectively. Meanwhile, the proposed method achieves the same accuracy compared with the homomorphic encryption based PPCF method and outperforms the randomized perturbation based PPCF method by approximately 0.13%–1.07% in accuracy.

The rest of this paper is organized as follows: Section 2 discusses related work. Section 3 presents the proposed privacy-preserving item-based collaborative filtering algorithm. Section 4 discusses and proves the privacy-preservation property of the proposed method. Section 5 presents and discusses the detailed evaluation results. Finally, we conclude this paper in Section 6.

2. Related work

Recommender systems have become an important research area in recent years [16]. Compared with content-based recommendation approach [17], collaborative filtering (CF) is one of the most widely adopted recommendation approach in existing recommender systems [16]. A wide range of CF methods have been proposed in the literature, which generally fall into two main categories: user-based CF [3,18], item-based CF [1,2,19]. Existing studies show that item-based CF methods could achieve comparable or better recommendation accuracy compared with user-based CF methods [19,20]. Meanwhile, item similarities can be calculated on a subset of user ratings [1], so that item-based CF methods are of better scalability. Moreover, user-based CF methods suffer from the "cold start user" problem, which is less of an issue in item-based CF methods. Overall, item-based CF methods play an important role in recommender system, so that the design of privacy-preserving item-based CF algorithm in this paper is beneficial.

Privacy issues of collaborative filtering have also been identified and investigated by recent works [5,9–13]. Existing works toward privacy-preserving collaborative filtering (PPCF) could be classified into two main categories. The first type of PPCF methods adopt cryptography to hide user private data. Canny [5] proposed a privacy-preserving SVD-based collaborative filtering method. In his solution, users compute the singular value decomposition (SVD) of the user–item matrix using homomorphic encryption, in which user privacy are protected by the encryption technique. After SVD computation, users can obtain recommendations via local computations. Aïmeur et al. [9] proposed the Alambic system, which can protect user privacy in a hybrid recommender system. The basic idea of Alambic system is that user private data are separated between the service provider and a semi-trusted third party, and the public key infrastructure is adopted to ensure data security. Thus, user privacy could be protected if the service provider does not collude with the semi-trusted third party. Kikuchi et al. [10] proposed a privacy-preserving collaborative filtering method, in which user similarities are calculated using homomorphic encryption. Meanwhile, item recommendation scores are also calculated using homomorphic encryption and then decrypted by a set of trusted authorities, so that user can obtain recommendations without privacy violation. Cryptography based PPCF solutions have the same accuracy compared with CF methods without privacy protection. However, encryption operations and computations on encrypted data greatly increase the computation overhead of recommender system. Our study using the Netflix Prize dataset shows that homomorphic encryption based PPCF method requires approximately 30X computation time compared with CF method without privacy protection. Thus, this kind of methods are not appropriate for applications with large-scale users and items. On the contrary, the privacy-preserving item-based CF method proposed in this paper does not rely on cryptography to protect user privacy, so that much higher efficiency is achieved.

The second type of PPCF methods adopt data perturbation techniques to inject noise on user private data before sending to recommender system, so that user privacy could be protected. Polat et al. [11] proposed a randomized perturbation technique to protect user privacy, in which random noises are injected to user rating data to prevent the recommender system from obtaining user privacy. However, the noise would affect the recommendation accuracy as demonstrated in their experiments. Zhang et al. [12] found that service provider could infer true user–item ratings from perturbed user–item ratings if all users are using the same perturbation variance. They proposed a two-way communication privacy-preserving approach, in which users perturb their item ratings based on the guidance from the recommender server. Their experimental results demonstrated that the new perturbation approach could reveal less privacy compared with existing perturbation approach at the same recommendation accuracy level. McSherry et al. [13] adopted the differential privacy method in collaborative filtering, which can hide true user–item ratings

with bounded probability of inferring from the perturbed data and computation results. As shown in their experiments, the accuracy losses range from approximately 2% to 14% with different amount of available data. The data perturbation based PPCF methods are as efficient as CF methods without privacy protection. But accuracy is the ultimate goal of recommender system, so that degradations in accuracy is not acceptable in real online services. Moreover, the data perturbation technique could not protect user privacy with strong guarantee because the service provider could derive user privacy from perturbed user data using machine learning techniques as demonstrated in recent works [14,15]. Compared with these data perturbation based PPCF methods, the privacy-preserving item-based CF method proposed in this paper does not manipulate user data or recommendation algorithm, so that there is no tradeoff between accuracy and privacy.

3. Efficient privacy-preserving item-based collaborative filtering

In this work, user privacy are protected by a proposed efficient secure multi-party computation (SMPC) protocol. In this section, we first present how to achieve unsynchronized SMPC in a distributed environment. Then, privacy-preserving item-based collaborative filtering using the proposed SMPC protocol is presented in detail.

3.1. Unsynchronized secure multi-party computation

The general goal of secure multi-party computation is to achieve the computation of n private values held by n parties ($n > 1$) without revealing the private value of each party during the computation. Secure multi-party computation was first studied by Yao [21], and later extended by Goldreich [22]. Most existing secure multi-party computation protocols require that all parties should be online and collaborate together to jointly compute a value [21,22]. However, in real online applications, the requirements of users being online simultaneously cannot be always guaranteed. To address this issue, we propose an unsynchronized secure multi-party computation protocol—*UnsyncSum* in this section, which can achieve jointly computations even when users are not online simultaneously.

Assume that there are n users, each user u_i holds a private value v_i, the goal of the proposed *UnsyncSum* protocol is to compute $\sum_i v_i$ without revealing each v_i to any of the other parties. In the *UnsyncSum* protocol, each of the private value v_i is randomly divided into *segment*s $S_i = \{s_1, \ldots, s_i\}$, such that $\sum_{s \in S_i} s = v_i$. Then, each user randomly sends the *segment*s to different users. Since the *segment*s are distributed among users, such that no single user can obtain all the *segment*s of a private value. And any subset of *segment*s does not reveal any information about the private value, so that the private value of each user could be protected. After this, each user computes the summation of its *segment*s and its received *segment*s, and sends the summation to the recommender server. As the summation of each user is a combination of multiple users' *segment*s, no privacy of individual user could be obtained in each summation. Finally, the server can compute the summation of all received values, which is equal to the summation of the original private values of all users. Please note that users are not required to be online simultaneously in the *UnsyncSum* protocol, but each user is required to be online once to participate in the protocol during the computation process. The detailed procedure of the proposed unsynchronized SMPC protocol—*UnsyncSum* is presented in Algorithm 1.

Now, we prove that the proposed *UnsyncSum* protocol (Algorithm 1) can obtain the correct summation in the following theorem.

Algorithm 1 UnsyncSum(U, V)

Require: U is a set of users, each of which holds a private value and wants to jointly compute the summation of all the values. V is the set of values held by users in U.

1: **while** not all users have participated **do**
2: **if** $u_i \in U$ gets online for the first time **then**
3: u_i locally divides its value v_i into a set of real-valued *segment*s $S_{u_i} = \{s_1, s_2, \ldots, s_{r_{u_i}}\}$ ($r_{u_i} \geq 3$ is a random number chosen by u_i) ensuring that $\sum_{1 \leq j \leq r_{u_i}} s_j = v_i$;
4: u_i randomly selects $r_{u_i} - 1$ online users as U_i; (Please note that a user may be selected multiple times if the number of online users is less than $r_{u_i} - 1$.)
5: **while** $U_i \neq \emptyset$ **do**
6: u_i randomly chooses *segment* $s \in S_{u_i}$ and user $u' \in U_i$, then sends s to u';
7: $U_i = U_i - \{u'\}$;
8: $S_{u_i} = S_{u_i} - \{s\}$;
9: **end while**
10: **if** u_i receives *segment* s from another user **then**
11: $S_{u_i} = S_{u_i} \cup \{s\}$;
12: **end if**
13: Before u_i gets offline, u_i computes $t_i = \sum_{s \in S_{u_i}} s$ and sends t_i to a randomly chosen online user;
14: **end if**
15: **end while**
16: The recommender server notifies users to terminate the protocol;
17: Each online user u_i computes $v_i' = \sum_{s \in S_{u_i}} s$ and sends v_i' to the recommender server;
18: Once all v_i's are received, the recommender server can compute $sum = \sum_i v_i'$;

Theorem 3.1. *For a set of users U, each user $u_i \in U$ holds a private value v_i, then running UnsyncSum protocol on U can obtain the correct summation $\sum_i v_i$.*

Proof. Let A be an $n \times n$ matrix (n is the number of users in U), and $A_{i,j}$ denotes the summation of segments that user u_i sent to user u_j, where $A_{i,j} = 0$ means u_i did not send any segment to u_j. And $A_{i,i}$ is the summation of all segments that are generated by u_i but did not send to any other user. Since the summation of all segments of u_i will equal to v_i, we have $v_i = \sum_i A_{i,j}$. When the server requests users to report their values, each online user u_i will correctly compute $v_i' = \sum_j A_{j,i}$ and report v_i'. Then, the server will have

$$\sum_i v_i' = \sum_i \left(\sum_j A_{j,i} \right) = \sum_i \sum_j A_{i,j} = \sum_i v_i.$$

Thus, we can conclude that the correct summation $\sum_i v_i$ is obtained. □

Meanwhile, the protocol is privacy-preserving in the semi-honest model [22], in which all parties follow the protocol properly except that they can infer the privacy of other parties based on intermediate values. The privacy of users are protected by the random *segment*s distribution, in which a user's private value is shared among no less than two users, so that no one can recover the private value based on only part of the *segment*s. Formal proof is given to prove the privacy-preservation property of the protocol in Section 4.

In the proposed *UnsyncSum* protocol, the computation and communication complexities are both $O(1)$ per user if we consider the size of random parts as a constant. Meanwhile, the server only need to receive numbers from users and then compute the

summation of these numbers, so the server-side computation and communication complexities are both $O(n)$, where n is the number of users. Since the computation and communication complexities are both linear in the number of users, we can say that the proposed *UnsyncSum* protocol is rather efficient.

3.2. Privacy-preserving item-based collaborative filtering using SMPC

Item-based collaborative filtering is a popular recommendation technique proposed by Amazon [1], and later adopted by many online services, such as Youtube [2]. In item-based CF, item similarities/correlations are first discovered. Then, item recommendations are generated based on user–item ratings and item correlations. Generally, item-based collaborative filtering algorithms work as follows [1,19]:

1. The recommender system first computes similarities/correlations among items pairs;
2. For a target user u, the recommender system finds items that are similar to items which were rated by u before;
3. For each target item i, the recommender system computes a weighted average to predict user u's rating on i.

In this section, we present how to achieve privacy-preserving item-based collaborative filtering using the proposed *UnsyncSum* protocol.

3.2.1. Privacy-preserving item similarity computation

In item-based CF algorithm, the key step is to compute similarities/correlations among items pairs. In this section, the *PrivateCosine* algorithm and the *PrivatePearson* algorithm are proposed, which can efficiently compute cosine similarity and Pearson correlation among item pairs while protecting the privacy of all users. Please note that, cosine similarity and Pearson correlation are two of the most commonly adopted similarity/correlation measures in item-based CF methods [1,19,16]. Other similarity/correlation measures, such as adjusted cosine similarity [19], Jaccard similarity [16], etc., could be computed similarly.

1. Privacy-preserving cosine similarity computation. In vector-space model, each item is described as a vector. The cosine value between two vectors can be considered as a measure of the similarity between the two items. The cosine similarity between item i and item j is computed as follows:

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\sum_{u\in U} r_{u,i} r_{u,j}}{\sqrt{\sum_{u\in U} r_{u,i}^2}\sqrt{\sum_{u\in U} r_{u,j}^2}} \tag{1}$$

where U is the set of users who have rated on item i or j, and $r_{u,i}$ is user u's rating on item i.

Here, a privacy-preserving algorithm—*PrivateCosine* is proposed to compute the cosine similarity between two items. In the *PrivateCosine* algorithm, the users first run *UnsyncSum* protocol to compute the three summations in Eq. (1). Then, after obtaining each of the summations in Eq. (1), the recommender server can compute the cosine similarity on the server side. The detailed procedure of proposed *PrivateCosine* is presented in Algorithm 2.

2. Privacy-preserving Pearson correlation computation. In statistical model, each item is described as a variable. Thus, the degree of dependent between two variables can be adopted as a measure of the correlation between the two items. The Pearson correlation between item i and item j is computed as follows:

$$sim(i,j) = corr_{i,j} = \frac{\sum_{u\in U}(r_{u,i}-\bar{r}_i)(r_{u,j}-\bar{r}_j)}{\sqrt{\sum_{u\in U}(r_{u,i}-\bar{r}_i)^2}\sqrt{\sum_{u\in U}(r_{u,j}-\bar{r}_j)^2}} \tag{2}$$

Algorithm 2 PrivateCosine(U, i, j)

Require: U is the set of users who have rated on item i or item j.

1: **for** each $u \in U$ **do**
2: u computes $r_{u,i}r_{u,j}$, $r_{u,i}^2$, and $r_{u,j}^2$ locally;
3: **end for**
4: Users in U run the *UnsyncSum* protocol to compute $\sum_{u\in U} r_{u,i}r_{u,j}$, $\sum_{u\in U} r_{u,i}^2$, and $\sum_{u\in U} r_{u,j}^2$;
5: The recommender server computes $\cos(i,j)$ as shown in Equation 1;

where U is the set of users who both rated item i and item j, $r_{u,i}$ is user u's rating on item i, and $\bar{r}_i$ is the average rating of item i.

Different from the cosine similarity, the Pearson correlation requires to compute the average ratings of items as shown in Eq. (2). Thus, the proposed *PrivatePearson* algorithm first needs to compute the average ratings of items using the *UnsyncSum* protocol. Then, the rest of the computations is conducted similarly as in *PrivateCosine*. The detailed procedure of the proposed *PrivatePearson* is presented in Algorithm 3.

Algorithm 3 PrivatePearson(U, i, j)

Require: U is the set of users who have rated on both item i and item j.

1: Users in U run the *UnsyncSum* protocol to compute $\sum_{u\in U} r_i$, $\sum_{u\in U} r_j$, and $\sum_{u\in U} \mathbb{1}(u,i,j)$. ($\mathbb{1}(u,i,j) = 1$ if u has rated both i and j, otherwise $\mathbb{1}(u,i,j) = 0$.)
2: $\bar{r}_i = \frac{\sum_{u\in U} r_i}{\sum_{u\in U}\mathbb{1}(u,i,j)}$, $\bar{r}_j = \frac{\sum_{u\in U} r_j}{\sum_{u\in U}\mathbb{1}(u,i,j)}$;
3: **for** each $u \in U$ **do**
4: u computes $(r_{u,i}-\bar{r}_i)(r_{u,j}-\bar{r}_j)$, $(r_{u,i}-\bar{r}_i)^2$, and $(r_{u,j}-\bar{r}_j)^2$ locally;
5: **end for**
6: Users in U run the *UnsyncSum* protocol to compute $\sum_{u\in U}(r_{u,i}-\bar{r}_i)(r_{u,j}-\bar{r}_j)$, $\sum_{u\in U}(r_{u,i}-\bar{r}_i)^2$, and $\sum_{u\in U}(r_{u,j}-\bar{r}_j)^2$;
7: The recommender server computes $corr_{i,j}$ as shown in Equation 2;

3.2.2. Privacy-preserving item recommendation generation

After the similarities/correlations among item pairs are generated, the recommender server sends these similarities/correlations to users. Then, users locally compute item recommendation scores using the weighted sum technique [1,19] as follows:

$$\tilde{r}_{u,i} = \frac{\sum_{j\in I_i} sim(i,j) * r_{u,j}}{\sum_{j\in I_i} sim(i,j)} \tag{3}$$

where I_i denotes the set of items that are similar to item i. As user ratings on items are stored locally, item similarities are obtained from the server, so that the local item recommendation computation will not violate user privacy.

3.2.3. Model updating

In item-based collaborative filtering, one common challenge is how to efficiently update the item similarity model when more user–item rating data are incrementally available. Here, we propose two efficient incremental methods to update the item similarity models.

1. Incremental updating for cosine similarity. The incremental updating for cosine similarity requires the computation of $\sum_{u\in U'} r_{u,i}r_{u,j}$, $\sum_{u\in U'} r_{u,i}^2$ and $\sum_{u\in U'} r_{u,j}^2$ (U' is the new set of users who have rated item i or item j) using the proposed *UnsyncSum* protocol. Let U be the set of users in the original cosine similarity computation, then all users in U are included in U'. Thus, we

do not need to add the data of users in U. We just need to run the *UnsyncSum* protocol on users in $U' - U$, and obtain the update for each summation. Then, the recommender server could add these updates to the original $\sum_{u\in U} r_{u,i}r_{u,j}$, $\sum_{u\in U} r_{u,i}^2$ and $\sum_{u\in U} r_{u,j}^2$, and recompute the cosine similarity as Eq. (1).

2. Incremental updating for Pearson correlation. The incremental updating for Pearson correlation is more challenging, because the average ratings of items change as new item ratings are obtained. Thus, the original computation results could not be utilized directly as the case for cosine similarity. Next, we describe how to update the Pearson correlation incrementally and efficiently.

Let $\bar{r}_i$ and $\bar{r}'_i$ be the average ratings of item i on the original user–item rating data and new user–item rating data, respectively. Let δ_i be the difference between the two average ratings of item i, so we have

$$\bar{r}'_i = \bar{r}_i + \delta_i. \tag{4}$$

Then, we discuss the updating for $\sum_{u\in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)$, $\sum_{u\in U}(r_{u,i} - \bar{r}_i)^2$, and $\sum_{u\in U}(r_{u,j} - \bar{r}_j)^2$ one by one hereinafter.

Let $\sum_{u\in U'}(r_{u,i} - \bar{r}'_i)(r_{u,j} - \bar{r}'_j)$ be the new numerator of Eq. (2), where U' is the new set of users who have rated item i and item j. Then, we divide this expression into three parts as follows:

$$\begin{aligned}
&\sum_{u\in U'}(r_{u,i} - \bar{r}'_i)(r_{u,j} - \bar{r}'_j)\\
&= \sum_{u\in U}(r_{u,i} - \bar{r}'_i)(r_{u,j} - \bar{r}'_j) + \sum_{u\in U'-U}(r_{u,i} - \bar{r}'_i)(r_{u,j} - \bar{r}'_j)\\
&= \sum_{u\in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j) + \Delta_{i,j} + \sum_{u\in U'-U}(r_{u,i} - \bar{r}'_i)(r_{u,j} - \bar{r}'_j).
\end{aligned} \tag{5}$$

The first term of Eq. (5) could be obtained from the original Pearson correlation computation, and the third term of Eq. (5) could be computed as in *PrivatePearson*. Thus, the main difficulty is to compute $\Delta_{i,j}$, which can be computed as follows:

$$\begin{aligned}
\Delta_{i,j} &= \sum_{u\in U}(r_{u,i} - \bar{r}'_i)(r_{u,j} - \bar{r}'_j) - \sum_{u\in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)\\
&= \sum_{u\in U}(r_{u,i} - (\bar{r}_i + \delta_i))(r_{u,j} - (\bar{r}_j + \delta_j))\\
&\quad - \sum_{u\in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)\\
&= \sum_{u\in U}((r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j) - \delta_j(r_{u,i} - \bar{r}_i)\\
&\quad - \delta_i(r_{u,i} - \bar{r}_i) + \delta_i\delta_j) - \sum_{u\in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)\\
&= \sum_{u\in U}\delta_i\delta_j
\end{aligned} \tag{6}$$

where δ_i (δ_j) is the difference between initial average rating of item i (j) and new average rating of i (j).

Since $\sum_{u\in U}(r_{u,i} - \bar{r}_i)^2$ and $\sum_{u\in U}(r_{u,j} - \bar{r}_j)^2$ can be updated in similar fashion, so we only present how to incrementally update $\sum_{u\in U}(r_{u,i} - \bar{r}_i)^2$, which can be divided into three parts as follows:

$$\begin{aligned}
\sum_{u\in U'}(r_{u,i} - \bar{r}'_i)^2 &= \sum_{u\in U}(r_{u,i} - \bar{r}'_i)^2 + \sum_{u\in U'-U}(r_{u,i} - \bar{r}'_i)^2\\
&= \sum_{u\in U}(r_{u,i} - \bar{r}_i)^2 + \Delta_{i^2} + \sum_{u\in U'-U}(r_{u,i} - \bar{r}'_i)^2.
\end{aligned} \tag{7}$$

Similarly, the first term of Eq. (7) can be obtained in the original Pearson correlation computation, and the third term can be computed as in *PrivatePearson*. And the Δ_{i^2} can be computed as follows:

$$\begin{aligned}
\Delta_{i^2} &= \sum_{u\in U}(r_{u,i} - \bar{r}'_i)^2 - \sum_{u\in U}(r_{u,i} - \bar{r}_i)^2\\
&= \sum_{u\in U}(r_{u,i} - (\bar{r}_i + \delta_i))^2 - \sum_{u\in U}(r_{u,i} - \bar{r}_i)^2\\
&= \sum_{u\in U}((r_{u,i} - \bar{r}_i)^2 - 2\delta_i(r_{u,i} - \bar{r}_i) + \delta_i{}^2) - \sum_{u\in U}(r_{u,i} - \bar{r}_i)^2\\
&= \sum_{u\in U}\delta_i{}^2.
\end{aligned} \tag{8}$$

As we can see, updating of Pearson correlation relies on the updating of average item ratings. The average ratings of items can be updated incrementally, because the recommender server just need to obtain the number of new users and the summation of new ratings using the *UnsyncSum* protocol as in *PrivatePearson*. Then, $\sum_{u\in U}\delta_i\delta_j$, $\sum_{u\in U}\delta_i{}^2$, and $\sum_{u\in U}\delta_j{}^2$ can be computed efficiently on the server side. Thus, the Pearson correlation can be updated incrementally and efficiently on the server side.

3.3. Analysis

3.3.1. Complexity analysis

1. Complexity of item similarity computation. On the server side, the complexity of similarity computation between two items is $O(n)$, where n is the number of users. Thus, the total complexity for computing similarities among all item pairs is $O(n * m^2)$ (m is the number of items), because there are totally $\frac{1}{2}m(m-1)$ item pairs. This server-side complexity is similar to that in the non-privacy-preserving item-based CF methods [1,19]. However, in those methods, $3n$ multiplications and $3n$ additions are required to compute the similarity between two items. In our method, all multiplications are performed by users, only $3n$ additions are required. Thus, the proposed method is much more efficient than those methods on the server side. On the client side, the computation and communication complexities are both $O({m_u}^2)$ for each user u, where m_u is the number of items that u have rated. This is because each user only needs to participate in the similarity computation of items which were rated by him/her before. Generally, user only rates a small set of items, so that the client-side computation and communication overhead are low. Note that the client-side complexities of the proposed method are higher than those of non-privacy-preserving collaborative filtering methods, whose computation complexity is 0 and communication complexity is typically $O(m_u)$ [19]. But the proposed method is beneficial to users because they can preserve their privacy by only performing a small amount of extra communication and computation.

2. Complexity of item recommendation generation. In our method, all item recommendations are generated on the client side. For each user u, the computation complexity is $O(m_u)$ for recommending one item, where m_u is the number of items that u have rated. This is because one weighted sum on m_u items should be performed to compute the recommendation score of an item. As stated above, users generally rates a small set of items, so that the item recommendation generation is also efficient.

3. Complexity of model updating. In the updating for cosine similarity and Pearson correlation, the computation and communication complexities for updating the similarity between two items on the server side are both $O(n')$, where n' is the number of new users. Thus, the total complexity for updating similarities among all item pairs is $O(n' * m^2)$, where m is the number of items. On the client side, the computation and communication complexities are also $O({m_u}^2)$ for each new user u, where m_u is the number of items that u have rated. Please note that, users who have already participated in similarity computations do not need to perform any computation during the updating.

Table 1
Similarity computation complexity comparisons between the proposed methods and traditional methods.

	Cosine similarity	Pearson similarity
Traditional	$(3\alpha + 3\beta)n + 3\beta$	$(5\alpha + 3\beta)n + 3\beta$
Proposed (similarity computation)	$3\alpha n_{i,j} + 3\beta$	$3\alpha n_{i,j} + 3\beta$
Proposed (model updating)	$3\alpha n'_{i,j} + 6\beta$	$3\alpha n'_{i,j} + 6\beta$

3.3.2. Efficiency analysis and comparison

As analyzed above, the server-side complexities of item similarity computation for the proposed methods are similar to those of non-privacy-preserving item-based CF methods. However, it should be noted that the proposed methods can distribute a large fraction of computation to users, so that the server-side computation overhead is significantly reduced.

1. Efficiency analysis and comparison of similarity computation.

Firstly, the proposed privacy-preserving similarity computation methods can reduce the time of data preparation. As shown in Eqs. (1) and (2), both $\sum_{u\in U} r_{u,i}r_{u,j}$ and $\sum_{u\in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)$ require pair-wise data $r_{u,i}$ and $r_{u,j}$. This implies that data preparation is required for traditional item similarity computation, which has $O(1)$ complexity for each pair-wise multiplication and $O(n)$ complexity for the summation over all pair-wise multiplications (n is the number of users). In contrast, the proposed Algorithm 2 shows that $r_{u,i}r_{u,j}$, $r^2_{u,i}$ and $r^2_{u,j}$ are computed by users in a distributed fashion, and similarly for $(r_{u,i}-\bar{r}_i)(r_{u,j}-\bar{r}_j)$, $(r_{u,i}-\bar{r}_i)^2$ and $(r_{u,j}-\bar{r}_j)^2$ in Algorithm 3. Thus, data preparation is avoided in the proposed methods.

Secondly, the proposed methods require less server-side computation. In Algorithm 2, $r_{u,i}r_{u,j}$, $r^2_{u,i}$ and $r^2_{u,j}$ are computed by users, and similarly for $(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)$, $(r_{u,i} - \bar{r}_i)^2$ and $(r_{u,j} - \bar{r}_j)^2$ in Algorithm 3. Thus, this part of computation is avoided on the server side. Meanwhile, the *UsyncSum* protocol only needs to compute the summation of values from online users rather than the whole set of users, which further reduces the server-side computation overhead. To precisely assess the computation efficiencies of the proposed methods, we assume each floating-point addition/subtraction operation consumes α CPU cycles and each floating-point multiplication/division operation consumes β CPU cycles. Let n be the total number of users in the system and $n_{i,j}$ be the number of online users when the server computes the similarity between item i and item j. Then, the detailed computation efficiencies of the proposed methods and the traditional item similarity computation methods are listed in Table 1. Since $\beta > \alpha$ [23] and $n \gg n_{i,j}$, we can conclude that the server-side computation overheads of the proposed similarity computation methods are much lower than those of the traditional methods.

2. Efficiency analysis and comparison of model updating.

Model updating is required for both our method and non-privacy-preserving CF methods, such as [1] and [19], when new ratings become available after item-to-item similarity model has been obtained. To the best of our knowledge, no incremental method for updating item-to-item similarity model have been proposed before. The incremental updating approach proposed in our method is much more efficient than other updating approaches that re-compute the similarities among all items.

As shown in Table 1, the overall server-side computation complexities for updating the similarity between an item pair (i, j) are both $3\alpha n'_{i,j} + 6\beta$ for Cosine similarity and Pearson similarity ($n'_{i,j}$ is the number of online users when updating the similarity between item i and item j). Since $\beta > \alpha$ and $n \gg n'_{i,j}$, the server-side complexities for model updating are much lower than updating by re-computing the similarities among all item pairs (row one in Table 1). For the client side, the model updating has the same complexity as the proposed similarity computation methods, which are both $O(\Delta m^2_u)$ per user (Δm_u is the number of items that u has newly rated).

4. Discussion

4.1. Secure multi-party computation

To prove that the proposed item-based collaborative filtering is privacy-preserving, we adopt the privacy definition in secure multi-party computation. We first discuss the privacy-preservation property of the proposed method under the semi-honest model [22], in which all parties follow the computation protocol properly except that they can infer the privacy of other parties based on intermediate values. Later, privacy under malicious model is discussed in detail. The formal definition of private multi-party computation in the semi-honest model is adopted from Goldreich's work [22], quoted below:

Definition 1 (*Privacy w.r.t. Semi-honest Behavior [22]*)**.**

- $f : (0,1^*)^m \mapsto (0,1^*)^m$ be an m-ary function, and $f_i(x_1, \ldots, x_m)$ denotes the ith element of $f(x_1, \ldots, x_m)$.
- For $i = \{i_1, \ldots, i_t\} \subset [m] \stackrel{def}{=} \{1, \ldots, m\}$, $f_I(x_1, \ldots, x_m)$ denotes the subsequence $f_{i_1}(x_1, \ldots, x_m), \ldots, f_{i_t}(x_1, \ldots, x_m)$.
- π is an m-party protocol for computing f.
- $VIEW_i^{\pi}(\bar{x})$ is the *View* of the ith party during an execution of π on $\bar{x} = (x_1, \ldots, x_m)$.
- $VIEW_I^{\pi}(\bar{x}) \stackrel{def}{=} (I, VIEW_{i_1}^{\pi}(\bar{x}), \ldots, VIEW_{i_t}^{\pi}(\bar{x}))$, for $I = \{i_1, \ldots, i_t\}$.

We say that π privately computes f if there exists a polynomial-time algorithm, denoted S, such that $\{(S(I, (x_{i_1}, \ldots, x_{i_t}), f_I(\bar{x})), f(\bar{x}))\}_{\bar{x}\in(0,1^*)^m} \stackrel{def}{\equiv} \{VIEW_I^{\pi}(\bar{x}), OUTPUT^{\pi}(\bar{x})\}_{\bar{x}\in(0,1^*)^m}$ for every I as shown above, where $OUTPUT^{\pi}(\bar{x})$ denotes the output sequence of all parties during the execution represented in $VIEW_I^{\pi}(\bar{x})$.

The above privacy definition states that a multi-party computation protocol is privacy-preserving if the view of each party during the execution of the protocol could be simulated by a polynomial-time algorithm knowing only the input and the output of the party.

Another key theory that we adopt to prove the privacy-preservation property of the proposed CF method is the Composition Theorem under semi-honest model (Theorem 4.1). Detailed proof of Theorem 4.1 could be found in [22], and thus is omitted here.

Theorem 4.1 (*Composition theorem for the semi-honest model [22]*)**.** *Suppose that g is privately reducible to f and that there exists a protocol to privately compute f. Then there exists a protocol to privately compute g.*

4.2. Privacy preservation in semi-honest model

Based on Definition 1 and Theorem 4.1, we first prove that each component of the proposed item-based collaborative filtering method is privacy-preserving in this section. Then, based on the Composition Theorem, we can conclude that the proposed item-based CF method is privacy-preserving.

4.2.1. Privacy preservation of UnsyncSum protocol

In the proposed item similarity/correlation computation, the *UnsyncSum* protocol is proposed to protect user privacy. We first prove that the proposed *UnsyncSum* protocol is privacy-preserving in the semi-honest model in the following theorem.

Theorem 4.2. *Given a set of users U ($|U| > 1$), each user $u_i \in U$ holds a private value v_i. The proposed UnsyncSum protocol can privately compute $\sum_i v_i$ in the semi-honest model.*

Proof. We construct a simulator to simulate the stages of the *UnsyncSum* protocol as follows:

- *Stage* 1: In this stage, each user u_i randomly divides its private value v_i into r_{u_i} *segment*s. The simulator for u_i can run exactly as what u_i performs in real computation. Since each *segment* of v_i is randomly distributed in $\mathbf{R}$, so that it is indistinguishable from what other user views in real computation. Thus, the outputs of u_i in this stage are successfully simulated.
- *Stage* 2: In this stage, u_i randomly sends $r_{u_i} - 1$ *segment*s of v_i to $r_{u_i} - 1$ users. The simulator for u_i can randomly select *segment*s from S_{u_i} to simulate the output of u_i. Again, since each *segment* of v_i is randomly distributed in $\mathbf{R}$, so that it is indistinguishable from what other user views in real computation. Thus, the outputs of u_i in this stage are successfully simulated.
- *Stage* 3: In this stage, u_i sends $\sum_{s\in S_{u_i}} s$ to the recommender server or another online user. The simulator for u_i can just simulate $\sum_{s\in S_{u_i}} s$ as the output of u_i, because this value is also a random number.
- *Stage* 4: In this stage, the recommender server computes the summation of all received values. Since there is no communication in this stage, the simulator for each user does not need to simulate anything.

The above simulator is linear in the size of the input/output of each user, which means that a polynomial-time simulator is successfully constructed. Thus, the *UnsyncSum* protocol can privately compute $\sum_i v_i$. □

4.2.2. Privacy preservation of item similarity computation

Since the multi-party summation in item similarity computation are achieved by the proposed *UnsyncSum* protocol, so that the privacy preservation property of item similarity computation can be easily proved. The formal proofs are presented in the following Theorems 4.3 and 4.4 for cosine similarity computation and Pearson correlation computation, respectively.

Theorem 4.3. *Given two items i and j, the proposed PrivateCosine algorithm can privately compute* $\cos(i, j)$.

Proof. In *PrivateCosine*, each user u first computes $r_{u,i}r_{u,j}$, $r_{u,i}^2$, and $r_{u,j}^2$ locally, so that user privacy are preserved. Then, $\sum_{u\in U} r_{u,i}r_{u,j}$, $\sum_{u\in U} r_{u,i}^2$, and $\sum_{u\in U} r_{u,j}^2$ are computed using the *UnsyncSum* protocol. As proved above, the *UnsyncSum* protocol can privately compute summations, so these computations are privacy-preserving. At last, the recommender server computes $\cos(i, j)$ using $\sum_{u\in U} r_{u,i}r_{u,j}$, $\sum_{u\in U} r_{u,i}^2$, and $\sum_{u\in U} r_{u,j}^2$, in which no privacy of individual user are revealed. Thus, based on Theorem 4.1, we can conclude that the *PrivateCosine* algorithm can privately compute $\cos(i, j)$. □

Theorem 4.4. *Given two items i and j, the proposed PrivatePearson algorithm can privately compute* $corr_{i,j}$.

Proof. This theorem can be similarly proved as in Theorem 4.3, so the proof is omitted here. □

4.2.3. Privacy preservation of item recommendation

In the proposed privacy-preserving item-based CF method, the item similarity computations are proved to be privacy-preserving, and the item recommendation generations are conducted on the client side, so that the privacy preservation property of the proposed method can be easily proved using the composition theorem. Thus, formal proofs are omitted here.

4.3. Privacy protection in malicious model

The previous section has discussed the privacy preservation property of the proposed item-based CF method in the semi-honest model. Actually, the proposed method has stronger privacy guarantee than the semi-honest model. Considering malicious model, in which users could manipulate their inputs/outputs or collude to attack a target user, the proposed method is privacy-preserving except that all other users are colluding to attack a target user. Otherwise, malicious users cannot attack the target user at all, because *segment*s of the target user's private value can be sent to someone who is not colluding with the malicious users. For other cases, in which collusion is not happening, single malicious user could only disrupt the results, but cannot learn any private information that are not revealed by the computation results. Since it is not practical that all users in the system are colluding to attack a target user, so that the proposed method can provide strong privacy guarantee even facing with malicious adversaries.

5. Experimental results

In this section, we evaluate the proposed privacy-preserving item-based collaborative filtering algorithm using the Netflix Prize dataset. In Sections 3 and 4, the privacy-preserving item-based collaborative filtering is formally described and proved. Therefore, the following quantitative studies focus on recommendation efficiency and accuracy.

- **System efficiency** is measured by the overall computation time for recommending all items to users on the server side.
- **Recommendation accuracy** is measured by MAE (Mean Average Error) and RMSE (Root Mean Square Error), which are defined as follows:

$$\mathrm{MAE} = \frac{1}{n}\sum_{u=1}^{n} |r_{u,i} - \tilde{r}_{u,i}|,$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{u=1}^{n} (r_{u,i} - \tilde{r}_{u,i})^2} \tag{9}$$

where $\tilde{r}_{u,i}$ is the predicted rating of user u on item i, and $r_{u,i}$ is the real rating from the dataset. It should be noted that for both MAE and RMSE, the smaller value indicates better recommendation accuracy, but RMSE is more sensitive to large errors than MAE.

All the experiments are conducted on the Netflix Prize dataset, which consists of 17,770 movies, 480,189 users, and almost 100 million known ratings in the scale from 1 to 5. In the experiments, the number of items vary from 1777 (10% of items) to 17,770 (all items).

In the experiments, the proposed algorithm is compared against two well-known privacy-preserving collaborative filtering (PPCF) solutions. The first method is a randomized perturbation based PPCF solution (RP) proposed by Polat et al. [11], in which random noises are injected to user rating data to prevent the recommender server from obtaining user privacy. The second method is a homomorphic encryption based PPCF solution (HE) proposed by Kikuchi et al. [10], in which user similarities and recommendation scores are calculated using homomorphic encryption. All the experiments are conducted using Ali cloud service environment (www.aliyun.com), and each server node is equipped with Xeon E5-2430 dual CPU and 16 GB memory.

5.1. System efficiency comparison

Since the server-side efficiency is the bottleneck of recommender system [16], this experiment compares the server-side efficiencies of the three methods. Fig. 1 shows the computation

Table 2
MAE comparison of the proposed method and the RP method.

	Item Size				
Method	$m = 1777$	$m = 3554$	$m = 5321$	$m = 7108$	$m = 8885$
Proposed	0.7670	0.7656	0.7613	0.7587	0.7582
RP ($\gamma = 50\%$)	0.7707	0.7676	0.7626	0.7595	0.7591
RP ($\gamma = 95\%$)	0.7943	0.7800	0.7704	0.7644	0.7634
	Item Size				
Method	$m = 10662$	$m = 12439$	$m = 14216$	$m = 15993$	$m = 17770$
Proposed	0.7579	0.7572	0.7565	0.7559	0.7551
RP ($\gamma = 50\%$)	0.7584	0.7578	0.7569	0.7563	0.7555
RP ($\gamma = 95\%$)	0.7608	0.7608	0.7586	0.7575	0.7563

Table 3
RMSE comparison of the proposed method and the RP method.

	Item Size				
Method	$m = 1770$	$m = 3540$	$m = 5310$	$m = 7080$	$m = 8850$
Proposed	0.9747	0.9674	0.9599	0.9559	0.9548
RP ($\gamma = 50\%$)	0.9788	0.9698	0.9614	0.9568	0.9558
RP ($\gamma = 95\%$)	1.0129	0.9881	0.9731	0.9640	0.9629
	Item Size				
Method	$m = 10620$	$m = 12390$	$m = 14160$	$m = 15930$	$m = 17700$
Proposed	0.9545	0.9537	0.9526	0.9515	0.9502
RP ($\gamma = 50\%$)	0.9551	0.9544	0.9531	0.9519	0.9505
RP ($\gamma = 95\%$)	0.9592	0.9589	0.9558	0.9539	0.9519

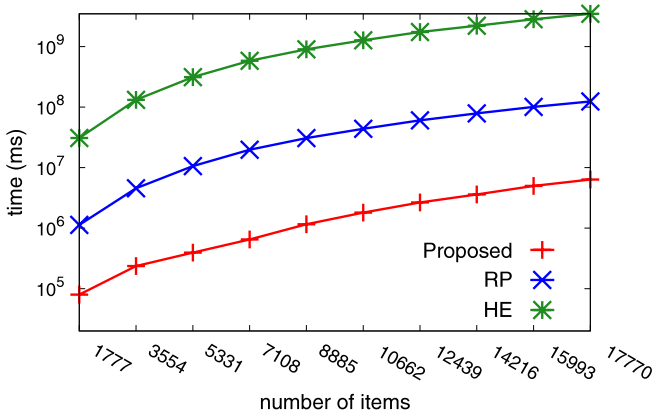

Fig. 1. Computation time comparison of the proposed method, HE, and RP.

time of the three methods on the server side. (In our method, we assume that all users are online, which increases the computation overhead compared with applying our method in real systems.) As we can see, the proposed method can greatly reduce the computation overhead on the server side. The proposed method consistently outperforms the RP method and HE method by over 14X and 386X across different dataset sizes, respectively, in recommendation efficiency. Note that, the RP method is of the same computation efficiency as some non-privacy-preserving item-based CF methods, such as the item-based CF methods proposed in [1,19]. This indicates that the proposed privacy-preserving item-based CF is even more efficient than some of the non-privacy-preserving item-based CF methods on the server side.

The reason why the proposed method could achieve high efficiency is that only addition operations are required on the server side during similarity computation. Meanwhile, the recommendation score computations are performed by clients, which further reduces the computation overhead on the server side. In the RP method, all the computations are performed on the server side, so it is not as efficient as the proposed method. In the HE method, all computations are performed on encrypted data, which greatly increases the computation overhead of the recommender server.

5.2. Recommendation accuracy comparison

In recommendation accuracy comparison, the proposed method and the HE method have no accuracy loss, so the same accuracies could be achieved in the two methods. Thus, the following experiments focus on comparisons between the proposed method and the RP method. In the RP method, the range of random noises would have great impact on recommendation accuracy, larger noise would results in greater loss in accuracy but better privacy protection. In this experiment, we compare two different ranges of random noises: $[-0.67, 0.67]$ and $[-1.95, 1.95]$, in which $\gamma = 50\%$ or $\gamma = 95\%$ of noises fall into those ranges, respectively, in standard normal distribution [11]. Please note that the RP method has slightly different accuracies in different runs, because the noise vary in different runs. Thus, in this experiment, we run ten times for the RP method and use the average MAE and RMSE as the final results.

Tables 2 and 3 show the MAE and RMSE comparisons of the proposed method and the RP method in different settings. From the results, we can see that the proposed method achieves better recommendation accuracy. Compared with the RP method, the proposed method can reduce MAE by 0.14% and 0.94% on average and reduce RMSE by 0.13% and 1.07% on average when γ is 50% and 95%, respectively. Note that, the accuracy loss of the RP method decreases as m (the number of items) increases, which is because larger m can yield more accurate approximation for similarity estimation in the RP method. However, in the RP method, the noise on user–item ratings would affect the accuracy of both similarity computation and recommendation score computation, so that the recommendation accuracy is affected. When the range of random noises grows larger, i.e., stronger privacy guarantee could be achieved, more accuracy losses are observed in the RP method. On the contrary, the proposed method does not need to trade accuracy for privacy.

6. Conclusion

This paper presents an efficient algorithm for privacy-preserving item-based collaborative filtering, which can protect user privacy during recommendation process without compromising

recommendation accuracy and efficiency. In the proposed method, item similarities/correlations are incrementally computed using a proposed unsynchronized secure multi-party computation protocol. After that, recommendations are generated on the client side with privacy. The proposed method is evaluated on the Netflix Prize dataset, and experimental results demonstrate that the proposed method outperforms a randomized perturbation based PPCF solution and a homomorphic encryption based PPCF method by over 14X and 386X, respectively, in recommendation efficiency. Meanwhile, the proposed method achieves the same accuracy as the homomorphic encryption based PPCF method, and outperforms the randomized perturbation based PPCF method by approximately 0.13%–1.07% in accuracy.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61233016, 61332008 and 61272533, the National Science Foundation under awards CNS-0910995 and CNS-1162614, and the Shanghai Science & Technology Committee Project under Grant Nos. 11JC1400800 and 13ZR1401900.

References

[1] G. Linden, B. Smith, J. York, Amazon.com recommendations: Item-to-item collaborative filtering, IEEE Internet Comput. 7 (1) (2003) 76–80.
[2] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, He Yu, Mike Lambert, Blake Livingston, Dasarathi Sampath, The YouTube video recommendation system, in: Proceedings of the fourth ACM conference on Recommender systems, RecSys '10, ACM, 2010, pp. 293–296.
[3] S.Das Abhinandan, Datar Mayur, Garg Ashutosh, Rajaram. Shyam, Google News personalization: scalable online collaborative filtering, in: Proceedings of the 16th international conference on World Wide Web, WWW '07, ACM, 2007, pp. 271–280.
[4] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, Anmol N. Sheth, TaintDroid: An Information-flow tracking system for realtime privacy monitoring on smartphones, in: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, OSDI '10, 2010, pp. 1–6.
[5] John Canny, Collaborative filtering with privacy, in: Proceedings of 2002 IEEE Symposium on Security and Privacy, S&P '02, IEEE, 2002, pp. 45–57.
[6] Wondracek Gilbert, Holz Thorsten, Kirda Engin, Kruegel. Christopher, A practical attack to deanonymize social network users, in: Proceedings of 2010 IEEE Symposium on Security and Privacy, S&P '10, IEEE, 2010, pp. 223–238.
[7] Yuan Mingxuan, Chen Lei, S.Yu. Philip, Personalized privacy protection in social networks, In Proceedings of the VLDB Endowment 4 (2) (2010) 141–150.
[8] Dongsheng Li, Qin Lv, Huanhuan Xia, Li Shang, Tun Lu, Ning Gu, Pistis: A Privacy-preserving content recommender system for online social communities, in: Proceedings of 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT '11, pp. 79–86, 2011.
[9] Esma Aïmeur, Gilles Brassard, José Manuel Fernandez, Flavien Serge Mani Onana, Alambic: a privacy-preserving recommender system for electronic commerce, Int. J. Inf. Secur. 7 (5) (2008) 307–334.
[10] Hiroaki Kikuchi, Hiroyasu Kizawa, Minako Tada, Privacy-preserving collaborative filtering schemes, in: International Conference on Availability, Reliability and Security, ARES '09, IEEE, 2009, pp. 911–916.
[11] Huseyin Polat, Wenliang Du, Privacy-preserving collaborative filtering using randomized perturbation techniques, in: Proceedings of The Third IEEE International Conference on Data Mining, ICDM '03, IEEE, 2003, pp. 625–628.
[12] Sheng Zhang, James Ford, Fillia Makedon, A privacy-preserving collaborative filtering scheme with two-way communication, in: Proceedings of the 7th ACM Conference on Electronic Commerce, EC '06, 2006, pp. 316–323.
[13] Frank McSherry, Ilya Mironov, Differentially private recommender systems: building privacy into the net, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, ACM, 2009, pp. 627–636.
[14] Zhengli Huang, Wenliang Du, Biao Chen, Deriving private information from randomized data, in: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, SIGMOD '05, ACM, 2005, pp. 37–48.
[15] Sheng Zhang, James Ford, Fillia Makedon, Deriving private information from randomly perturbed ratings, in: Proceedings of the Sixth SIAM International Conference on Data Mining, SDM '06, SIAM, 124(59), 2006.
[16] Gediminas Adomavicius, Alexander Tuzhilin, Toward the next generation of recommender systems: A survey of the State-of-the-Art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749.
[17] Marko Balabanović, Yoav Shoham, Fab: Content-based, collaborative recommendation, Commun. ACM 40 (1997) 66–72.
[18] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, John Riedl, An algorithmic framework for performing collaborative filtering. in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99, 1999, pp. 230–237.
[19] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, John Reidl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, WWW'01, ACM, 2001, pp. 285–295.
[20] Manos Papagelis, Dimitris Plexousakis, Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents, Eng. Appl. Artif. Intell. 18 (7) (2005) 781–789.
[21] Andrew C. Yao, Protocols for secure computations, in: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, FOCS'82, IEEE, 1982, pp. 160–164.
[22] Oded Goldreich, Secure Multi-Party Computation. Final (incomplete) Draft, Version 1.4. 2002.
[23] Intel Corporation. Intel 64 and IA-32 Architectures Optimization Reference Manual. http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html. March 2014.

Dongsheng Li received the B.E. degree from University of Science and Technology of China, Hefei, China, in 2007, and the Ph.D. degree in School of Computer Science from Fudan University, Shanghai, China, in 2012. He is currently a Postdoctoral Researcher with the Department of Computer Science and Technology, Tongji University, Shanghai, China. His current research interests include recommender systems, online social networks, and smart grid.

Chao Chen received his Bachelor degree in Software Engineering from Zhejiang University of Technology, Zhejiang, China, in 2012. He is currently a graduate student with the Department of Computer Science and Technology, Tongji University, Shanghai, China. His research interests include privacy-preserving recommender systems.

Qin Lv received the B.E. degree (Hons.) from Tsinghua University, Beijing, China, in 2000, and the Ph.D. degree in Computer Science from Princeton University, Princeton, NJ, in 2006. She is currently an Assistant Professor with the Department of Computer Science, University of Colorado, Boulder. She has published more than 40 papers in peer-to-peer networks, large-scale similarity searches, air quality sensing, PHEV driving studies, and event modeling and recommendation in online social communities. Her current research interests include search systems, data mining, mobile systems, social networks, and data management.

Li Shang (S'99–M'04) received the B.E. degree (Hons.) from Tsinghua University, Beijing, China, and the Ph.D. degree from Princeton University, Princeton, NJ. He is currently an Associate Professor with the Department of Electrical, Computer, and Energy Engineering, University of Colorado, Boulder. He has authored or co-authored over 100 publications in computer systems, mobile computing and design for high-performance information systems. Dr. Shang currently serves as an Associate Editor of the IEEE Transactions on Very Large Scale Integration Systems and the ACM Journal on Emerging Technologies in Computing Systems. He was a recipient of the Best Paper Award in IEEE/ACM DATE 2010 and IASTED PDCS 2002. His work on FPGA power modeling and analysis was selected as one of the 25 Best Papers from FPGA. His work on temperature-aware on-chip networks was selected for publication in the MICRO Top Picks 2006. His work was a recipient of the Best Paper Award nominations at ISLPED 2010, ICCAD 2008, DAC 2007, and ASP-DAC 2006. He was a recipient of the Provost's Faculty Achievement Award in 2010 and his department's Best Teaching Award in 2006. He was a recipient of the NSF CAREER Award.

Yingying Zhao is currently a Ph.D. candidate with the Department of Computer Science and Technology, Tongji University, Shanghai, China. Her research interest includes privacy-preservation in information systems, big data analysis and information system design in smart grid.

Tun Lu graduated from Sichuan University, China with a B.Eng. in 2000, a M.Eng. in 2003 and a Ph.D. in 2006, all in Computer Science. He is now an Associate Professor at the School of Computer Science, Fudan University, China. His current research interests include collaborative computing, social computing and service computing.

Ning Gu received the Ph.D. degree in Computer Science from the Institute of Computing Technology, Chinese Academy of Sciences, China, 1995. He is a Professor and the Director of the Cooperative Information and Systems Lab at the School of Computer Science, Fudan University, China. His current research interests include computer-supported cooperative work, data management, distributed systems, and social networking. More information about his research is available at http://cscw.fudan.edu.cn/. He is a member of the IEEE.