Data Sci. Eng. (2016) 1(3):161–177
DOI 10.1007/s41019-016-0020-2

# A Practical Privacy-Preserving Recommender System

**Shahriar Badsha**[1] · **Xun Yi**[1] · **Ibrahim Khalil**[1]

Received: 13 July 2016 / Revised: 9 September 2016 / Accepted: 13 September 2016 / Published online: 28 September 2016


**Abstract** The main goal of a personalized recommender system is to provide useful recommendations on various items to the users. In order to generate recommendations, the service needs to access various types of user data such as previous product purchasing history, demographic and biographical information. However, users are sensitive to disclosure of personal information as it can be easily misused by malicious third parties. Consequently, there are unavoidable security concerns which will become known through attempted unauthorized access while providing the recommendation services. In order to protect against breaches of personal information, it is necessary to obfuscate the user information by means of an efficient encryption technique while simultaneously generating the recommendation by making true information inaccessible to the system. To address these challenges, we propose a privacy-preserving recommender system using homomorphic encryption, by which the system can provide recommendations without knowing the actual ratings. Our approach is based on the ElGamal cryptosystem by which both addition and multiplication of plaintexts can be performed. The performance of the proposed scheme shows significantly high accuracy in-terms of computation and communication costs as well as outperforming other existing solutions.

**Keywords** Data privacy · Recommender systems · Homomorphic encryption

✉ Shahriar Badsha
shahriar.badsha@rmit.edu.au

[1] RMIT University, Melbourne, VIC 3001, Australia

## 1 Introduction

Recommender systems [1] provide meaningful and useful recommendations to users by making use of explicit and implicit information about user preferences. Recommendations are also often based on the degree of similarity between the active user and all other users, or one particular item that the user has rated and all other items. The items can be of any type: books, movies, web pages, restaurants, sightseeing places, online news, and even lifestyles. By collecting information about users' preferences for different items, a recommender system creates their profiles. These preferences can help the recommender system to predict other items that might also be of interest to the user in the future. Content-based filtering (CBF) and collaborative filtering (CF) are the most commonly used techniques that generate recommendations for users based on their preferences. CBF predicts a user's rating on a particular item based on the previous ratings and item features, while CF generates recommendations based on the previous ratings only. In order to run the process of recommendations, users' profiles must be available to the recommender server (or service providers). Therefore, there are risks that such information is leaked to malicious parties which can lead to severe damage to the user's privacy (e.g. exposure or generating false recommendations) [2]. Figure 1 shows the general architecture of a conventional recommender system and possible ways in which privacy breaches can occur. It is thus crucial to adequately protect privacy of information managed by recommender systems. Existing approaches can be categorized as follows.

*Perturbation* In data perturbation methods, noises are injected to users' private data before sending the data to the server for generating recommendations. Zhang et al. [3]

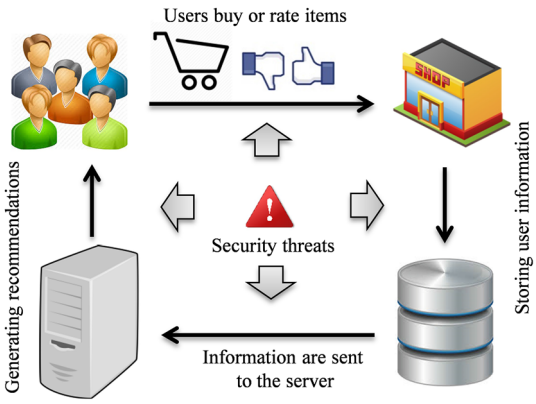


**Fig. 1** Traditional recommender system and its possible security concerns

proposed two data reconstruction methods based on Singular Value Decomposition and K-means clustering. These derive original private information from perturbed data in existing perturbation-based CF schemes. Zhang et al. also showed that the data perturbation techniques might not be able to secure the data properly. Another singular value decomposition (SVD) based on CF with privacy has been proposed [4] which maintains acceptable accuracy for a recommendation while protecting the privacy of user information based on randomized perturbation-based schemes. A distributed mechanism for users in CF was proposed [5] by which the users' profiles were made secure to untrusted server from accessing the data. Also, a trade-off between privacy and recommendation accuracy has been minimized in this technique.

*Differential Privacy* Little research has been carried out on differentially private recommender systems which also consider minimizing the trade-off between security and recommendation accuracy. A method [6] based on differential privacy has been proposed where the algorithm has been factored into two parts: an aggregation/learning phase that can be performed with differential privacy guarantees, and an individual recommendation phase that uses the learned correlations and an individual's data to provide personalized recommendations. Another example of a differential private recommender system was proposed by Machanavajjhala et al. [7] who presented a graph link-based recommendation and formalized the trade-off between privacy and accuracy. Extra noises are induced in these types of methods.

*Homomorphic Encryption* The homomorphic encryption techniques usually adopt cryptography to hide users' private data. Canny [8] proposed a scheme for privacy-preserving CF which reduces the filtering task to an iterative calculation of the aggregate, requiring only the addition of vectors of user data using homomorphic encryption techniques. This approach allows sums of encrypted vectors to be computed without exposing individual rating information of users. Another research [9] was conducted by the same author to protect an individual's personal data only. This approach was conducted based on a probabilistic factor analysis model. Its use was also suggested for different kinds of statistical analyses. Erkin et al. [10] proposed a collaborative filtering-based recommender system which uses efficient protocols based on homomorphic encryption and secure multiparty communication. They also applied the approach in a social trust network [11] aiming to achieve similar outcomes to previous work where the additional overhead with regard to computation and communication is minimized by packing data. Kikuchi et al. [12] addressed the large overhead in performing the cryptographic operations which is proportional to the number of users and items. Therefore, they came up with reducing the computation and communication costs by introducing clustering-based CF where users and items are grouped together based on their similarity. Tada et al. [13] proved that the cost for generating recommendation can be reduced by using item-based similarity instead of user-based. Therefore, the issue with scalability in the recommender system was handled in their proposed work where privacy was preserved using Paillier cryptosystem.

The main disadvantages of perturbation and differential privacy-based methods are as follows. Firstly, they suffer from poor quality in selecting the neighbours (similar users or items) due to inducing large noise. Secondly, they are not highly guaranteed in-terms of providing rigorous security. They also suffer from a trade-off between privacy and recommendation accuracy. Additionally, the existing homomorphic-based approaches experience high computational costs [14].

*Our Contributions* In this paper, we propose a new privacy-preserving recommender system, which allows the computations required for recommendations in a distributed manner and preserves user privacy without compromising recommendation accuracy and efficiency. We introduce the privacy protocol by ElGamal encryption which is based on public key cryptosystem. The main advantage of this cryptosystem is that it is semantically secure and allows certain types of computations on the ciphertexts. We assume a semi-trusted server named "recommender server" whose task is to perform the computations for recommendation on encrypted data. We propose different privacy protocols for item average and similarity computations as well as recommendations generations by which the privacy of users is preserved. Specifically, our main contributions are:

1. An efficient privacy-preserving item-based recommender system to protect user privacy during recommendation process.

2. Privacy-preserving item average and similarity computation protocols to calculate averages and similarities among the items without compromising user ratings. Moreover, which items have been rated are also hidden during these processes.
3. Two different types of solutions to generate recommendations securely: CBF and CF-based recommendations.

In summary, our proposed model is able to work as follows:

- Firstly, all users participate to compute average ratings of items. Users encrypt their rating as well as flag information to hide which items are actually rated. Specifically, users encrypt all ratings including zeros and send the ciphertexts to the server. The server computes averages using homomorphic properties and all users jointly decrypt the results.
- Secondly, to calculate the similarity among the items, all users locally perform certain computations and encrypt them. The server computes similarity among the items securely and allows all users to decrypt the results without revealing any private information.
- Finally, based on average ratings, similarities and target user's encrypted rating information, the "recommender server" computes recommendation scores homomorphically. The target user decrypts these ciphertexts using own private key and chooses highest recommended item from the results.

The remainder of this paper is organized as follows. In Sect. 2, we provide some preliminary studies related to recommender system and ElGamal cryptosystems. Section 3 represents the proposed methodology to compute item average, similarity and recommendation securely. A numerical example with a small user-item rating matrix has been provided to explain our proposed model clearly in Sect. 4. Sections 5 and 6, respectively, represent the security and performance analysis of our proposed model. The last section presents a conclusion to the overall study.

## 2 Preliminaries

There are two basic entities that drive a recommender system: *users* who use the recommender system to provide opinions as well as receive recommendations and *items* that are rated by users. The inputs to a recommender system are usually arithmetic rating values, which express the users' opinions of items and follow a specified numerical scale (example: 1: bad to 5: excellent). The outputs of a recommender system can be either predictions or recommendations.

Let $U = u_1, u_2, \ldots, u_n$ be the set of all $n$ users in a recommender system and $I = i_1, i_2, \ldots, i_m$ be the set of items where $m$ is the total number of items. Let $R$ be a rating matrix where $r_{i,j}$ is a rating provided by user $u_i$ on item $i_j$. Usually, the rating matrix is sparse because of missing values as it is not possible to rate all items nominated by all users in a system. The missing rating is denoted by $r_{i,j} = 0$. Hence, the primary goal of a recommender system is to predict the rating for a user $u_i$ on item $i_j$ which the user has not previously rated.

### 2.1 Similarity Calculation

To generate recommendations, one of the key steps is to calculate similarities/correlations among the item pairs. Cosine similarity is one of the commonly adopted similarity measures to determine the nearest neighbour in recommendation generation. Recalling the notations mentioned previously, the similarity between two items is defined as

$$s(i_j, i_k) = \frac{\sum_{i=1}^{n} r_{i,j} r_{i,k}}{\sqrt{r_{1,j}^2 + \cdots + r_{n,j}^2}\sqrt{r_{1,k}^2 + \cdots + r_{n,k}^2}} \tag{1}$$

where $i_j$ and $i_k$ represent two individual items. $r_{i,j}$ and $r_{i,k}$ represent the ratings provided by the user $u_i$ on those two items and $n$ represents the total number of users.

### 2.2 Recommendation Generation

Depending on different scenarios, the types of recommendations for users may vary. In our model, we consider two types of recommendations, by which the solutions represent determining recommendation scores for all items and based on these scores target user is able to choose the most suitable item for him.

#### *2.2.1 CBF-Based Recommendations*

In CBF, the recommendations are generated based on the items' features. The process is to check for similarity among the items which is calculated using item features first, and then, based on those similarity, the CBF generates recommendations for the target user. The equation for predicting the recommendation using CBF is:

$$P_{i,k} = \frac{\sum_{j=1}^{m} r_{i,j} \cdot s(i_j, i_k)}{\sum_{j=1}^{m} s(i_j, i_k)} \tag{2}$$

where $P_{i,k}$ denotes the rating prediction for user $u_i$. $k = \{1, 2, \ldots, m\}$ is the number of items that the target user has requested for the recommendation. $r_{i,j}$ and $s(i_j, i_k)$ denote the rating vector of user $u_i$ and similarity between item $i_j$ and $i_k$ respectively. Please note that in our proposed model

we use user ratings instead of item features to calculate similarity among the items.

#### 2.2.2 CF-Based Recommendations

Unlike the CBF, the CF-based algorithm generates recommendations based on item ratings. CF also works in two steps: first calculating similarity among the items based on their ratings and then predicting new ratings for the items. Given similarities among the items, the item-based CF computes the prediction of user $u_i$ for item $i_k$ by

$$P_{i,k} = \frac{R_k \cdot \sum_{j=1}^{m} s(i_k, i_j) + \sum_{j=1}^{m} (r_{i,j} - R_j) \cdot s(i_k, i_j)}{\sum_{j=1}^{m} s(i_k, i_j)} \tag{3}$$

where $P_{i,k}$ denotes the prediction for user $u_i$ on item $i_k$. $r_{i,j}$ and $s(i_k, i_j)$ represent the rating provided on item $i_j$ by user $u_i$ and the similarity between items $i_k$ and $i_j$ respectively. $R_k$ and $R_j$ represent the average ratings of items $i_k$ and $i_j$ respectively. Note that the average rating of a particular item is computed as dividing the total rating by the total number of users who have actually rated that item.

### 2.3 Homomorphic Encryption

The ElGamal cryptosystem presented in [15] is additively and multiplicatively homomorphic. This means that there exist two operations over the ciphertexts $E(M_1)$ and $E(M_2)$ such that the results of those operations correspond to new ciphertexts whose decryption yield the sum and multiplication of the plain texts $M_1$ and $M_2$.

$$\begin{aligned} E(M_1)E(M_2) &= E(M_1 + M_2) \\ E(M_1)^{M_2} &= E(M1 \cdot M_2) \end{aligned} \tag{4}$$

The ElGamal encryption scheme is a homomorphic and probabilistic public key encryption based on Diffie–Hellman key exchange. The ElGamal encryption scheme can be defined over any cyclic group $G$. Its security depends upon the difficulty of a certain problem in $G$ related to computing discrete logarithms. The ElGamal encryption scheme consists of three algorithms: the key generation, the encryption algorithm, and the decryption algorithm.

*Key Generation* The private and the public keys are $x \in Z_q$ and $y = g^x$ respectively, where $G$ is a cyclic group of order $q$ and $x = \{1, \ldots, q-1\}$.

*Encryption* A message $m$ is encrypted as follows.

$$(C_1, C_2) = (g^r, m \cdot y^r) \tag{5}$$

where $r$ is an arbitrary random number chosen by the encrypter. $C_1$ and $C_2$ are represented as ciphertexts.

*Decryption* The ciphertexts are decrypted by computing with the private key $x$ as follows.

$$\frac{C_2}{(C_1)^x} = \frac{m \cdot y^r}{(g^r)^x} = \frac{m \cdot y^r}{y^r} = m \tag{6}$$

*Homomorphic Property* Given two encryptions,

$$\begin{aligned} (C_{11}, C_{12}) &= (g^{r_1}, m_1 \cdot y^{r_1}) \\ (C_{21}, C_{22}) &= (g^{r_2}, m_2 \cdot y^{r_2}) \end{aligned} \tag{7}$$

the computation of multiplication for two ciphertexts is as follows:

$$\begin{aligned} (C_{11}, C_{12})(C_{21}, C_{22}) &= (C_{11}C_{21}, C_{12}C_{22}) \\ &= (g^{r_1} g^{r_2}, (m_1 y^{r_1})(m_2 y^{r_2})) \\ &= (g^{r_1 + r_2}, (m_1 m_2) y^{r_1 + r_2}) \end{aligned} \tag{8}$$

The resulting ciphertext is the encryption of $m_1 m_2$ which is a multiplication of two plaintexts. It has been shown that ElGamal is semantically secure, i.e. it is computationally infeasible to distinguish between the encryptions of any two given messages, if the decisional Diffie–Hellman problem is intractable [15].

## 3 Proposed Privacy-Preserving Recommender System

According to Fig. 2, our proposed scheme is mainly divided into two phases. Firstly, we provide privacy-preserving item average computation and privacy-preserving similarity computation as shown in Fig. 2a. Secondly, we represent privacy-preserving recommendations generation as shown in Fig. 2b. In the first phase, users encrypt their ratings and send the ciphertexts to the server. Server computes the averages as well as similarity among the items using homomorphic properties and all users collaborate to decrypt these ciphertexts. Afterwards, the similarity and averages are stored in server's database. According to Fig. 2b, only one user participates to get recommendations ('target user') and sends the ciphertexts of his item preferences to the server. The server computes recommendation scores homomorphically and sends the resultant ciphertexts to the user. The target user finally decrypts the ciphertexts using own private key. Two types of secure recommendation scores generation are shown: (1) CBF-based recommendations and (2) CF-based recommendations. Table 1 shows the mathematical notations and symbols used in later sections.

### 3.1 Privacy-Preserving Average and Similarity Computations

*Settings* Let a rating matrix shown in Table 2 consists of $n$ users who have provided ratings on $m$ items where each user and item are denoted as $u_i$ and $i_j$ respectively. The

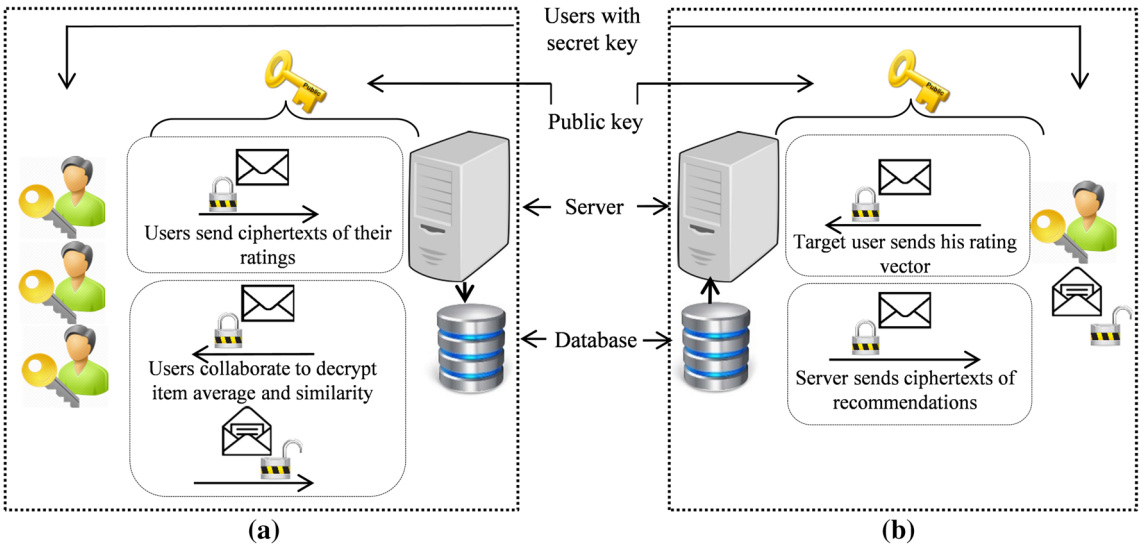


**Fig. 2** Framework of proposed privacy-preserving recommender system, divided into two phases: **a** *average and similarity computation*, all users participate and send the ciphertexts of their ratings to the server. Server performs homomorphic operation to calculate average and similarity, thereby stores the results in its own database, **b** *recommendations generation*, only one user participates and send the ciphertexts of own ratings. The server computes the recommendation using homomorphic properties and send the ciphertexts to the user. The decrypts the ciphertexts using own private key

**Table 1** Notations

| | |
|---|---|
| $E$ | Encryption |
| $(A, B)$ | Ciphertexts |
| $r$ | Random number |
| $G$ | Cyclic group |
| $g$ | Group generator |
| $y$ | Individual public key |
| $Y$ | Common public key |
| $x$ | Individual secret key |
| $r_{i,j}$ | Rating given by user $u_i$ on item $i_j$ |
| $R_j$ | Average rating of item $j$ |
| $M^{(a)}$ | $a$th message |
| $u_i$ | $i$th user |
| $i_j$ | $j$th item |
| $n$ | Total number of users |
| $m$ | Total number of items |
| $s(i_j, i_k)$ | Similarity between item $I_j$ and $i_k$ |
| $C_i$ | Decryption |
| $d^i$ | Results after discrete logarithm |
| $P$ | Rating prediction |
| $e$ | Modular exponentiation |
| $mul$ | Multiplication |
| $l$ | Size of the message |
| $f_{i,j}$ | Flags of ratings $r_{i,j}$ |

rating is denoted by $r_{i,j}$ meaning the user $u_i$ has given a rating on item $i_j$. We also assume that there is a secure channel established between the users and server to

**Table 2** User-item rating matrix

| Users/items | $i_1$ | $i_2$ | ... | ... | ... | ... | $i_m$ |
|---|---|---|---|---|---|---|---|
| $u_1$ | $r_{1,1}$ | ... | $r_{1,j}$ | ... | $r_{1,k}$ | ... | $r_{1,m}$ |
| $u_2$ | $r_{2,1}$ | ... | $r_{2,j}$ | ... | $r_{2,k}$ | ... | $r_{2,m}$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $u_i$ | $r_{i,1}$ | ... | $r_{i,j}$ | ... | $r_{i,k}$ | ... | $r_{i,m}$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $u_n$ | $r_{n,1}$ | ... | $r_{n,j}$ | ... | $r_{n,k}$ | ... | $r_{n,m}$ |

exchange any secret messages securely. To compute averages and similarity among the items securely, the server generates a cyclic group $G$, of large prime order $q$ with generator $g$. Each user $u_i$ randomly chooses secret key $x_i$ where $i = \{1, 2, \ldots, n\}$ and $x_i \in \{1, \ldots, q-1\}$. The public key is calculated by user $u_i$ as

$$y_i = g^{x_i} \tag{9}$$

Next, all users send their public keys $y_1, \ldots, y_i, \ldots, y_n$ to the server. For $n$ users, the server calculates a common public key $Y$ and broadcast it to all users to encrypt their ratings.

$$Y = y_1 \cdot y_2 \cdot \ldots \cdot y_n = \prod_{i=1}^{n} y_i = g^{\sum_{i=1}^{n} x_i} \tag{10}$$

### 3.1.1 Average Computation

Let the users denoted as $u_i = \{u_1, u_2, \ldots, u_n\}$ have participated with the server and encrypt their ratings $r_{i,j}$ as well as

flags $f_{i,j}$ (1, if there is a rating, 0 otherwise) using common public key $Y$. The server computes averages of items homomorphically, and the results are decrypted by collaboration of all users. The detailed steps are described below.

**Step 1** For all items, $i_j = i_1, i_2 \ldots, i_m$ each user $u_i$ encrypts their ratings and flags as $E(g^{r_{i,j}})$ and $E(g^{f_{i,j}})$ respectively. Then, they create the message $M_i^1$ as shown below, containing ciphertexts of their ratings and flags, which are sent to the server (note that, all users locally compute $g^m$ where $m$ denotes any message, using ElGamal encryption and $g^m$ is within the cyclic group $G$ ($g^m \in G$)).

$$M_i^1 = \left\{E(g^{r_{i,j}}), E(g^{f_{i,j}})\right\}_{(j=1,2,\ldots,m)} \tag{11}$$

where $i = \{1, 2, \ldots, n\}$. $r_{i,j}$ denotes a rating provided by user $u_i$ on item $i_j$ and flag $f_{i,j} = 1$ if there is any rating otherwise $f_{i,j} = 0$.

**Step 2** Using homomorphic property, the server computes the addition of the ratings and flags as

$$(A_{1,j}, B_{1,j}) = \prod_{i=1}^{n} E(g^{r_{i,j}}) \tag{12}$$

$$(A_{2,j}, B_{2,j}) = \prod_{i=1}^{n} E(g^{f_{i,j}}) \tag{13}$$

where $(A_{1,j}, B_{1,j})$ and $(A_{2,j}, B_{2,j})$ denote the ciphertexts of ratings and flags, respectively. To decrypt these ciphertexts, server broadcasts $A_{1,j}$ and $A_{2,j}$ to all users by

$$M^{(2)} = \{A_{1,j}, A_{2,j}\}_{(j=1,2,\ldots,m)}$$

**Step 3** For all users $u_i = \{u_1, u_2 \ldots, u_n\}$, they compute as follows using own private keys and send the following message to the server.

$$M_i^{(3)} = \left\{(A_{1,j})^{x_i}, (A_{2,j})^{x_i}\right\}_{(j=1,2,\ldots,m)} \tag{14}$$

**Step 4** The server decrypts the ciphertexts from Eqs. (12) and (13) by

$$C_{1,j} = \frac{B_{1,j}}{\prod_{i=1}^{n}(A_{1,j})^{x_i}} \tag{15}$$

$$C_{2,j} = \frac{B_{2,j}}{\prod_{i=1}^{n}(A_{2,j})^{x_i}} \tag{16}$$

Finally, the sum of the ratings and flags are computed by the server as

$$d_j^{(1)} = \log_g C_{1,j} \tag{17}$$

$$d_j^{(2)} = \log_g C_{2,j} \tag{18}$$

where $d_j^{(1)}$ and $d_j^{(2)}$ denote the sum of ratings and the sum of 1's (flags) of item $i_j$ respectively. Therefore, the average rating of item $i_j$ is calculated as follows.

$$R_j = \frac{d_j^{(1)}}{d_j^{(2)}} \tag{19}$$

*Remark 1* Since all users jointly decrypt the sum of ratings for each item as well as the number of users who have actually rated on that item, the server or any users pose no threats to any user's private key, personal ratings and flags. Recall that the server is semi-honest and does not disclose $d_j^{(1)}$, $d_j^{(2)}$ and $R_j$ to anyone.

**Theorem 1** *If all users and server follow the protocol, we have* $d_j^{(1)} = \sum_{i=1}^{n} r_{i,j}$, $d_j^{(2)} = \sum_{i=1}^{n} f_{i,j}$.

*Proof* From Eq. (12), the server computes the ciphertexts homomorphically as,

$$(A_{1,j}, B_{1,j}) = \prod_{i=1}^{n} E(g^{r_{i,j}}) = E\left(g^{\sum_{i=1}^{n} r_{i,j}}\right) \tag{20}$$

Using homomorphic property, server decrypts the above ciphertexts as follows:

$$\begin{aligned} C_{1,j} = \frac{B_{1,j}}{\prod_{i=1}^{n}(A_{1,j})^{x_i}} &= \frac{g^{r_{1,j}+\cdots+r_{n,j}} \cdot Y^{r_1+\cdots+r_n}}{\left(g^{r_1+\cdots+r_n}\right)^{x_1+\cdots+x_n}} \\ &= \frac{g^{r_{1,j}+\cdots+r_{n,j}} \cdot (y_1 \cdot \ldots \cdot y_n)^{r_1+\cdots+r_n}}{\left(g^{r_1+\cdots+r_n}\right)^{x_1+\cdots+x_n}} \\ &= \frac{g^{r_{1,j}+\cdots+r_{n,j}} \cdot \left(g^{x_1} \cdot \ldots \cdot g^{x_n}\right)^{r_1+\cdots+r_n}}{\left(g^{r_1+\cdots+r_n}\right)^{x_1+\cdots+x_n}} \\ &= \frac{g^{r_{1,j}+\cdots+r_{n,j}} \cdot \left(g^{x_1+\cdots+x_n}\right)^{r_1+\cdots+r_n}}{\left(g^{r_1+\cdots+r_n}\right)^{x_1+\cdots+x_n}} \\ &= g^{r_{1,j}+\cdots+r_{n,j}} = g^{\sum_{i=1}^{n} r_{i,j}} \end{aligned} \tag{21}$$

To get $\sum_{i=1}^{n} r_{i,j}$, the server computes discrete logarithm as

$$d_j^{(1)} = \log_g C_{1,j} = \log_g g^{\sum_{i=1}^{n} r_{i,j}} = \sum_{i=1}^{n} r_{i,j} \qquad \square$$

In the same way, we can get

$$d_j^{(2)} = \log_g C_{2,j} = \log_g g^{\sum_{i=1}^{n} f_{i,j}} = \sum_{i=1}^{n} f_{i,j} \qquad \square$$

*Remark 2* The value of $\sum_{i=1}^{n} r_{i,j}$ and $\sum_{i=1}^{n} f_{i,j}$ is not large, therefore computing discrete logarithm is not hard.

### 3.1.2 Similarity Computation

To calculate similarity among the items $s(i_j, i_k)$ where $i_j$ and $i_k$ represent two different items, we use cosine similarity measure as shown in Eq. (1). Similar to the average computation, this protocol is also based on homomorphic properties of public key cryptosystem. According to the

protocol each user $u_i$, where $i = \{1, 2, \ldots, n\}$ locally computes pairwise multiplication $r_{i,j} \cdot r_{i,k}$ and square $r_{i,j}^2$ of their preferences. Then, they encrypt these results using common public key $Y$ and send them to the server. Afterwards, the server performs similarity computation homomorphically and allows all users to decrypt the results. The detailed steps are described as follows.

**Step 1** To calculate the similarity among the items, all users $u_i$, where $u_i = u_1, u_2, \ldots, u_n$, locally computes $g^{r_{i,j} \cdot r_{i,k}}$ and $g^{r_{i,j}^2}$, and sends the message $M_i^{(4)}$ containing these ciphertexts as shown below.

$$M_i^{(4)} = \left\{ E(g^{r_{i,j} \cdot r_{i,k}}), E(g^{r_{i,j}^2}) \right\}_{(j,k=1,2,\ldots,m; k \geq j)} \tag{22}$$

where $i = \{1, 2, \ldots, n\}$. $r_{i,j} \cdot r_{i,k}$ and $r_{i,j}^2$ represent pairwise multiplications of two different items' ratings $i_j$ and $i_k$, square of each individual ratings on each item by user $u_i$.

**Step 2** The homomorphic product of the ciphertexts which are received by the server is computed as follows.

$$(A_{j,k}, B_{j,k}) = \prod_{i=1}^{n} E(g^{r_{i,j} \cdot r_{i,k}}) \tag{23}$$

$$(A_j, B_j) = \prod_{i=1}^{n} E(g^{r_{i,j}^2}) \tag{24}$$

where $j, k = \{1, 2, \ldots, m\}$ and $k \geq j$.

The server generates the message $M^{(5)}$ as shown below and broadcasts to all users for decryption.

$$M^{(5)} = \left\{ A_{j,k}, A_j \right\}_{(j,k=1,2,\ldots,m; k \geq j)}$$

**Step 3** After receiving the message $M^{(5)}$, all users $u_i = \{u_1, u_2, \ldots, u_n\}$ create new messages $M_i^{(6)}$ as shown below using their individual private key $x_i$.

$$M_i^{(6)} = \left\{ (A_{j,k})^{x_i}, (A_j)^{x_i} \right\}_{(j,k=1,2,\ldots,m; k \geq j)} \tag{25}$$

where $i = \{1, 2, \ldots, n\}$. The message $M_i^{(6)}$ is sent back to the server.

**Step 4** Once the above message is received by the server, it starts to decrypt the ciphertexts of Eqs. (23) and (24) as follows.

$$C_{3,j,k} = \frac{B_{j,k}}{\prod_{i=1}^{n} (A_{j,k})^{x_i}} \tag{26}$$

$$C_{4,j} = \frac{B_j}{\prod_{i=1}^{n} (A_j)^{x_i}} \tag{27}$$

The server finally determine the plaintexts by computing discrete logarithm as,

$$d_{jk}^{(3)} = \log_g C_{3,j,k} \tag{28}$$

$$d_j^{(4)} = \log_g C_{4,j} \tag{29}$$

where $j, k = \{1, 2, \ldots, m\}$ and $k \geq j$. $d_{jk}^{(3)}$ and $d_j^{(4)}$ are represented as decryption of pairwise product and square of ratings, respectively.

After extracting the plaintexts from above equations, the server computes the similarity between two items $i_j$ and $i_k$ using Eq. (1) as follows.

$$s(i_j, i_k) = \frac{d_{jk}^{(3)}}{\sqrt{d_j^{(4)}} \cdot \sqrt{d_k^{(5)}}} \tag{30}$$

where similar to the $d_j^{(4)}$, $d_k^{(5)}$ is also calculated for item $i_k$. Thus, the server generates $m \times m$ item similarity matrix and stores it in the database. Note that, since the server is semi-honest, it does not disclose any result from this computation.

**Theorem 2** *If all users and server follow the protocol, we have* $d_{jk}^{(3)} = g^{\sum_{i=1}^{n} r_{i,j} \cdot r_{i,k}}$ *and* $d_j^{(4)} = g^{\sum_{i=1}^{n} r_{i,j}^2}$.

*Proof* From Eq. (23) the server computes the ciphertexts using homomorphic property as

$$(A_{j,k}, B_{j,k}) = \prod_{i=1}^{n} E(g^{r_{i,j} \cdot r_{i,k}}) = E\left(g^{\sum_{i=1}^{n} r_{i,j} \cdot r_{i,k}}\right) \tag{31}$$

Therefore, from Eq. (26), the server decrypts the similarity by

$$\begin{aligned} C_{3,j,k} = \frac{B_{j,k}}{\prod_{i=1}^{n} (A_{j,k})^{x_i}} &= \frac{g^{r_{1,j} \cdot r_{1,k} + \cdots + r_{n,j} \cdot r_{n,k}} \cdot Y^{r_1 + \cdots + r_n}}{(g^{r_1 + \cdots + r_n})^{x_1 + \cdots + x_n}} \\ &= \frac{g^{r_{1,j} \cdot r_{1,k} + \cdots + r_{n,j} \cdot r_{n,k}} \cdot (y_1 \cdot \ldots \cdot y_n)^{r_1 + \cdots + r_n}}{(g^{r_1 + \cdots + r_n})^{x_1 + \cdots + x_n}} \\ &= \frac{g^{r_{1,j} \cdot r_{1,k} + \cdots + r_{n,j} \cdot r_{n,k}} \cdot (g^{x_1} \cdot \ldots \cdot g^{x_n})^{r_1 + \cdots + r_n}}{(g^{r_1 + \cdots + r_n})^{x_1 + \cdots + x_n}} \\ &= \frac{g^{r_{1,j} \cdot r_{1,k} + \cdots + r_{n,j} \cdot r_{n,k}} \cdot (g^{x_1 + \cdots + x_n})^{r_1 + \cdots + r_n}}{(g^{r_1 + \cdots + r_n})^{x_1 + \cdots + x_n}} \\ &= g^{r_{1,j} \cdot r_{1,k} + \cdots + r_{n,j} \cdot r_{n,k}} \\ &= g^{\sum_{i=1}^{n} r_{i,j} \cdot r_{i,k}} \end{aligned} \tag{32}$$

To find the result of $\sum_{i=1}^{n} r_{i,j} \cdot r_{i,k}$, the server computes discrete logarithm as follows.

$$d_{jk}^{(3)} = \log_g C_{3,j,k} = \log_g g^{\sum_{i=1}^{n} r_{i,j} \cdot r_{i,k}} = \sum_{i=1}^{n} r_{i,j} \cdot r_{i,k} \qquad \square$$

Similarly, for all items $i_j = \{i_1, \ldots, i_m\}$ we can prove $d_j^{(4)} = \sum_{i=1}^{n} r_{i,j}^2$.

### 3.2 Proposed Privacy-Preserving Recommendation Generation

We provide two types of solutions for generating recommendations privately using Content-based Filtering (CBF) and Collaborative Filtering (CF). Let the target user $u_i$ has requested the list of recommendations for all items and sends his item preferences to the server. The server generates ciphertexts of recommendation scores homomorphically for all items. The target user finally decrypts these ciphertexts using own private key $x_i$ and thus gets the list of recommendations for all items.

*Settings* To generate the recommendations, we assume there is only one user $u_i$ (this could be a new user or from the set of $n$ users, given that he has provided ratings on same set of $m$ items) who has requested for recommendations. The user $u_i$ holds the public key $y_i$ to encrypt the ratings and private key $x_i$ to decrypt the ciphertexts. The server holds item–item similarity $s(i_j, i_k)$ and averages of items' ratings $R_j$ where $j = \{1, 2, \ldots, m\}$.[1]

#### 3.2.1 Privacy-Preserving CBF-Based Recommendations

To generate CBF-based recommendations, the protocol is divided into three main steps. Firstly, the target user encrypts his personal rating vector and sends the ciphertexts to server. Secondly, the server performs homomorphic operations and encryption to generate the ciphertexts of numerator and denominator of Eq. (2) (CBF-based method), respectively. Finally, the user receives these ciphertexts from server and decrypts them to determine recommendation scores. The detailed steps are described below.

**Step 1** For all items, where $i_j = i_1, i_2, \ldots, i_m$, the target user $u_i$ encrypts his ratings[2] using the individual public key $y_i$ and sends the message $M_i^{(7)}$ containing these ciphertexts as follows.

$$M_i^{(7)} = \{E(g^{r_{i,j}})\}_{j=1,2,\ldots,m}$$

[1] While in the both recommendation processes. users' personal ratings and recommendations are privacy sensitive, the item–item similarity matrix and averages are commercially valuable to the server. This information cannot be made public or sent to the user to generate recommendations since this will affect the service provider's business.

[2] Target user encrypts 0 s, while there is no rating and each of the ratings is multiplied by 100 since the similarities and averages are multiplied by 100 by the server to cope with homomorphic operations using ElGamal cryptosystem.

**Step 2**

1. After receiving message $M_i^{(7)}$, the server performs homomorphic operation to generate ciphertexts [numerator of Eq. (2)] by

$$(A_{2,k}, B_{2,k}) = \prod_{j=1}^{m} E(g^{r_{i,j}})^{s(i_k, i_j)} \tag{33}$$

   where $k = 1, 2, \ldots, m$.
2. For all items $j = \{1, 2, \ldots, m\}$ the server locally computes the encryption of $\sum_{j=1}^{m} s(i_k, i_j)$ [denominator of Eq. (2)] using target user $u_i$'s public key $y_i$ (recall that similarity among the items are already stored in server's database). Firstly, server locally computes $g^{\sum_{j=1}^{m} s(i_k, i_j)}$ and encrypts them as

$$(A_{3,k}, B_{3,k}) = E\left(g^{\sum_{j=1}^{m} s(i_k, i_j)}\right) \tag{34}$$

   The server sends the message $M^{(8)}$ containing these ciphertexts to user $u_i$ as follows.

$$M^{(8)} = \left\{(A_{2,k}, B_{2,k}), (A_{3,k}, B_{3,k})\right\}_{k=1,2,\ldots,m}$$

**Step 3** For all $k = 1, 2, \ldots, m$, using target user's private key $x_i$, these ciphertexts are decrypted by

$$C_{4,k} = \frac{B_{2,k}}{(A_{2,k})^{x_i}} \tag{35}$$

$$C_{5,k} = \frac{B_{3,k}}{(A_{3,k})^{x_i}} \tag{36}$$

Finally the user $u_i$ computes discrete logarithm to derive the results as

$$d_k^{(6)} = \log_g C_{4,k} \tag{37}$$

$$d_k^{(7)} = \log_g C_{5,k} \tag{38}$$

Therefore, the target user $u_i$ determines the recommendation by

$$P_{i,k} = \frac{d_k^{(6)}}{d_k^{(7)}} \tag{39}$$

where $P_{i,k}$ denotes the predicted recommendation on item $i_k$ where $k = 1, 2, \ldots, m$ and the item with highest prediction is finally recommended to the user.

**Theorem 3** *If all users and server follow the protocol, we have* $d_k^{(6)} = g^{\sum_{j=1}^{m} r_{i,j} \cdot s(k,j)}$ *and* $d_k^{(7)} = \sum_{j=1}^{m} s(i_k, i_j)$.

*Proof* From Eq. (33), the server homomorphically computes the ciphertexts by

$$(A_{2,k}, B_{2,k}) = \prod_{j=1}^{m} E(g^{r_{i,j}})^{s(i_k,i_j)} = E\left(g^{\sum_{j=1}^{m} r_{i,j}s(i_k,i_j)}\right) \quad (40)$$

Using Eq. (35), user $u_i$ decrypts the ciphertexts using his private key $x_i$ as follows.

$$C_{4,k} = \frac{A_{2,k}}{(B_{2,k})^{x_i}} = \frac{g^{\sum_{j=1}^{m} r_{i,j}s(i_k,i_j)} \cdot (g^{x_i})^{\sum_{j=1}^{m} r_i s(i_k,i_j)}}{\left(g^{\sum_{j=1}^{m} r_i s(i_k,i_j)}\right)^{x_i}} = g^{\sum_{j=1}^{m} r_{i,j}s(i_k,i_j)} \quad (41)$$

To get the value of $\sum_{j=1}^{m} r_{i,j}s(i_k, i_j)$ , the user computes discrete logarithm as

$$d_k^{(6)} = \log_g C_{4,k} = \log_g g^{\sum_{j=1}^{m} r_{i,j}s(i_k,i_j)} = \sum_{j=1}^{m} r_{i,j}s(i_k, i_j) \quad \square$$

Similarly, we can prove $d_k^{(7)} = \sum_{j=1}^{m} s(i_k, i_j)$.

### 3.2.2 Privacy-Preserving CF-Based Recommendations

Similar to the CBF-based recommendation process, the proposed private CF-based method[3] also works in three main steps, which include sending target user's encrypted ratings to server, performing homomorphic operations as well as encryption by the server and finally, determining the final recommendations by the target user. The detailed steps are as follows.

**Step 1** For all items $i_j = i_1, i_2, \ldots, i_m$, the target user $u_i$ encrypts his ratings and sends message $M_i^{(9)}$ as

$$M_i^{(9)} = \{E(g^{r_{i,j}})\}_{j=1,2,\ldots,m}$$

**Step 2** In this step,[4] server computes the ciphertexts of numerator and denominator of Eq. (3) homomorphically and sends them to target user.

1. Firstly, the server encrypts $R_k \cdot \sum_{j=1}^{m} s(i_k, i_j) + \sum_{j=1}^{m}(r_{i,j} - R_j) \cdot s(i_k, i_j)$ (numerator of Eq. (3)). Before performing homomorphic operations, it locally computes $g^{R_k \cdot \sum_{j=1}^{m} s(i_k,i_j)}$, $g^{R_j}$ and encrypts them as $E(g^{R_k \cdot \sum_{j=1}^{m} s(i_k,i_j)})$ and $E(g^{R_j})$. Finally, the ciphertexts of numerator are generated by the server homomorphically as

$$(A_{4,k}, B_{4,k}) = E\left(g^{R_k \cdot \sum_{j=1}^{m} s(i_k,i_j)}\right) \cdot \left(\prod_{j=1}^{m}(E(g^{r_{i,j}})/E(g^{R_j}))^{s(i_k,i_j)}\right) \quad (42)$$

   where $k = 1, 2, \ldots, m$.
2. Secondly, to generate the ciphertexts of denominator, server locally computes $g^{\sum_{j=1}^{m} s(i_k,i_j)}$ and encrypts them by

$$(A_{5,k}, B_{5,k}) = E\left(g^{\sum_{j=1}^{m} s(i_k,i_j)}\right) \quad (43)$$

For all items $k = \{1, 2, \ldots, m\}$, the server sends message $M^{(10)}$ containing these ciphertexts to target user $u_i$ as follows.

$$M^{(10)} = \{(A_{4,k}, B_{4,k}), (A_{5,k}, B_{5,k})\}_{k=1,2,\ldots,m} \quad (44)$$

**Step 3** The target user $u_i$ receives this message and decrypts the ciphertexts using his own private key $x_i$ by

$$C_{6,k} = \frac{B_{4,k}}{(A_{4,k})^{x_i}} \quad (45)$$

$$C_{7,k} = \frac{B_{5,k}}{(A_{5,k})^{x_i}} \quad (46)$$

User $u_i$ computes discrete logarithm to retrieve the results as

$$d_k^{(8)} = \log_g C_{6,k} \quad (47)$$

$$d_k^{(9)} = \log_g C_{7,k} \quad (48)$$

Finally the rating prediction is calculated as follows.

$$P_{i,k} = \frac{d_k^{(8)}}{d_k^{(9)}} \quad (49)$$

where $P_{i,k}$ denotes the recommendation for user $u_i$ on item $i_k$ , where $k = \{1, 2, \ldots, m\}$. Once the user $u_i$ gets the list of recommendations for all items, he chooses the highest recommended item for him.

**Theorem 4** *If the target user and the server follow the protocol, we have* $d_k^{(8)} = R_k \cdot \sum_{j=1}^{m} s(i_k, i_j) + \sum_{j=1}^{m}(r_{i,j} - R_j) \cdot s(i_k, i_j)$ *and* $d_k^{(9)} = \sum_{j=1}^{m} s(i_k, i_j)$.

*Proof* Homomorphically, the server computes the ciphertexts from Eq. (42) as

$$(A_{4,k}, B_{4,k}) = E\left(g^{R_k \cdot \sum_{j=1}^{m} s(i_k,i_j)}\right) \cdot \left(\prod_{j=1}^{m}(E(g^{r_{i,j}})/E(g^{R_j}))^{s(i_k,i_j)}\right) = E\left(g^{R_k \cdot \sum_{j=1}^{m} s(i_k,i_j) + \sum_{j=1}^{m}(r_{i,j} - R_j) \cdot s(i_k,i_j)}\right) \quad (50)$$

The target user $u_i$ decrypts the ciphertexts using his own private key $x_i$ as,

[3] Note that, unlike the CBF, CF-based method includes one additional step which is subtracting the item's average rating from user's preference on that item.

[4] The server holds average ratings $R_j$ and similarities among the items $s(i_k, i_j)$.

**Table 3** Rating matrix

| $u_i/i_j$ | $i_1$ | $i_2$ | $i_3$ | $i_4$ |
|---|---|---|---|---|
| $u_1$ | 3 | 5 | 0 | 4 |
| $u_2$ | 0 | 1 | 5 | 0 |
| $u_3$ | 2 | 3 | 2 | 4 |
| $R_j$ | 2.5 | 3 | 3.5 | 4 |

$$\begin{aligned} C_{6,k} &= \frac{B_{4,k}}{(A_{4,k})^{x_i}} \\ &= \frac{g^{R_k \cdot \sum_{j=1}^{m} s(i_k,i_j) + \sum_{j=1}^{m}(r_{i,j}-R_j)\cdot s(i_k,i_j)} \cdot \left(g^{x_i}\right)^{r_i + \sum_{j=1}^{m} r_i \cdot s(i_k,i_j)}}{\left(g^{r_i + \sum_{j=1}^{m} r_i \cdot s(i_k,i_j)}\right)^{x_i}} \\ &= g^{R_k \cdot \sum_{j=1}^{m} s(i_k,i_j) + \sum_{j=1}^{m}(r_{i,j}-R_j)\cdot s(i_k,i_j)} \end{aligned} \tag{51}$$

To get $R_k \cdot \sum_{j=1}^{m} s(i_k, i_j) + \sum_{j=1}^{m}(r_{i,j} - R_j) \cdot s(i_k, i_j)$, the target user computes discrete logarithm as

$$\begin{aligned} d_k^{(8)} &= \log_g C_{6,k} = \log_g g^{R_k \cdot \sum_{j=1}^{m} s(i_k,i_j) + \sum_{j=1}^{m}(r_{i,j}-R_j)\cdot s(i_k,i_j)} \\ &= R_k \cdot \sum_{j=1}^{m} s(i_k, i_j) + \sum_{j=1}^{m}(r_{i,j} - R_j) \cdot s(i_k, i_j) \end{aligned}$$ □

In the same way, we can prove $d_k^{(9)} = \sum_{j=1}^{m} s(i_k, i_j)$.

*Remark 3* The target user may send the indices of items for which he has not provided any ratings. In this case, the server does not consider those items while computing the ciphertexts of recommendations.

### 3.3 Numerical Examples of Proposed Method

To describe the processes of proposed privacy-preserving average computation, item–item similarity calculation and recommendations generation clearly, we provide numerical examples with small $3 \times 4$ matrix shown in Table 3. From this table, the users and items are represented as $u_i = \{u_1, u_2, u_3\}$ and $i_j = \{i_1, i_2, i_3, i_4\}$ respectively. The numerical values in this matrix are denoted as ratings, given by the users on different items. Before starting these processes, the server generates a cyclic group $G$, of a large prime order $q$ with generator $g$. Users $u_i$ where $i = 1, 2, 3$, randomly choose their secret keys $x_i$, where $x_i \in \{1, \ldots, q-1\}$ and calculate public keys as $y_i = g^{x_i}$. In below, the detailed processes with numerical example are shown.

#### 3.3.1 Average Computation

In this example, we show how to compute average of user ratings on item $i_1$, denoted as $R_1$. Firstly, users $u_1$, $u_2$ and $u_3$ send their ratings and the flags for item $i_1$ as follows

$$M_1^{(1)} = \{E(g^3), E(g^1)\}$$
$$M_2^{(1)} = \{E(g^0), E(g^0)\}$$
$$M_3^{(1)} = \{E(g^2), E(g^1)\}$$

Once the server receives these messages of ratings and flags, it starts to compute products of the ciphertexts (sum of the ratings in plaintexts) by following.

$$(A_{1,1}, B_{1,1}) = E(g^3) \cdot E(g^0) \cdot E(g^2) = E(g^5)$$

and products of the ciphertexts of flags as

$$(A_{2,1}, B_{2,1}) = E(g^1) \cdot E(g^0) \cdot E(g^1) = E(g^2)$$

The server broadcasts the message $M^{(2)}$ containing a portion of the ciphertexts for all items to all users by

$$M^{(2)} = \{A_{1,1}, A_{2,1}\}$$

Users create the message $M_i^{(3)}$ and send them to server as follows

$$M_{i=1,2,3}^{(3)} = \left\{(A_{1,1})^{x_i}, (A_{2,1})^{x_i}\right\}$$

where $i = 1, 2, 3$.

Finally, the server decrypts the ciphertexts of ratings and flags as

$$C_{1,1} = \frac{B_{1,1}}{\prod_{i=1}^{3}(A_{1,1})^{x_i}} = g^5$$
$$C_{2,1} = \frac{B_{2,1}}{\prod_{i=1}^{3}(A_{2,1})^{x_i}} = g^2$$

Using discrete logarithm, the plaintexts of ratings and flags can be determined, respectively, as $d_1^{(1)} = \log_g g^5$ and $d_1^{(2)} = \log_g g^2$. Therefore, the average of item $i_1$ is computed by

$$R_1 = \frac{5}{2} = 2.5$$

Similarly, the averages are calculated for all other items as

$$R_2 = \frac{9}{3} = 3, \quad R_3 = \frac{7}{2} = 3.5, \quad R_4 = \frac{8}{2} = 4$$

As ElGamal cryptosystem can not handle fraction number, the server multiplies every plaintexts by 100. The average ratings of all items are shown in Table 3.

#### 3.3.2 Similarity Calculation

In this example, we show how to compute similarity between item $i_1$ and $i_2$ securely. Firstly, all users compute product of pairwise ratings and square of each individual rating locally, thereby encrypt the results. For instance, all

users send the message $M_i^{(4)}$ containing the ciphertexts for item $i_1$ and $i_2$ to the server as follows:

$$M_1^{(4)} = \left\{E(g^{3\cdot5}), E(g^{3^2}), E(g^{5^2})\right\}$$
$$M_2^{(4)} = \left\{E(g^{0\cdot1}), E(g^{0^2}), E(g^{1^2})\right\}$$
$$M_3^{(4)} = \left\{E(g^{2\cdot3}), E(g^{2^2}), E(g^{3^2})\right\}$$

Once these ciphertexts are received by the server, it computes new ciphertexts of similarity between item $i_1$ and $i_2$ homomorphically as

$$(A_{1,2}, B_{1,2}) = E(g^{3.5}) \cdot E(g^{0.1}) \cdot E(g^{2.3}) = E(g^{21})$$
$$(A_1, B_1) = E(g^{3^2}) \cdot E(g^{0^2}) \cdot E(g^{2^2}) = E(g^{13})$$
$$(A_2, B_2) = E(g^{5^2}) \cdot E(g^{1^2}) \cdot E(g^{3^2}) = E(g^{35})$$

where $(A_{1,2}, B_{1,2})$ represent the numerator and, $(A_1, B_1)$ and $(A_2, B_2)$ represent denominator of Eq. (1) (similarity calculation). Similarly, the server computes similarities between other pairs of items and sends a portion of these ciphertexts to all users as

$$M^{(5)} = \left\{A_{1,2}, A_1, A_2\right\}$$

All users collaborate to decrypt the ciphertexts send the message $M_i^{(6)}$ to server by

$$M_{i=1,2,3}^{(6)} = \left\{(A_{1,2})^{x_i}, (A_1)^{x_i}, (A_2)^{x_i}\right\}$$

where $x_i$ represents private key of each user and $i = 1, 2, 3$. The server decrypts above ciphertexts of similarities between item $i_1$ and $i_2$ as

$$C_{3,1,2} = \frac{B_{1,2}}{\left(\prod_{i=1}^{3} A_{1,2}\right)^{x_i}} = g^{21}$$
$$C_{3,1} = \frac{B_1}{\left(\prod_{i=1}^{3} A_1\right)^{x_i}} = g^{13}$$
$$C_{3,2} = \frac{B_2}{\left(\prod_{i=1}^{3} A_2\right)^{x_i}} = g^{35}$$

Using discrete logarithm, the final results are derived as, $d_{1,2}^{(3)} = \log_g g^{21} = 21$, $d_1^{(4)} = \log_g g^{13} = 13$ and $d_2^{(5)} = \log_g g^{35} = 35$. Finally, the similarity between items $i_1$ and $i_2$ is computed by the server as follows.

$$s(i_1, i_2) = \frac{21}{\sqrt{13} \cdot \sqrt{35}} = 0.99$$

Thus, the similarities among other items are calculated, and the resultant matrix is stored by the server. The range of the similarity between two items is $-1$ to 1, where they denote most dissimilarity and most similarity, respectively. The similarities among all items are shown in Table 4, where each similarity is multiplied by 100 to cope with homomorphic properties of ElGamal encryption.

**Table 4** Similarity among the items

| | $i_1$ | $i_2$ | $i_3$ | $i_4$ |
|---|---|---|---|---|
| $i_1$ | 100 | 99 | 20 | 98 |
| $i_2$ | | 100 | 35 | 95 |
| $i_3$ | | | 100 | 26 |
| $i_4$ | | | | 100 |

### 3.3.3 Recommendation Generation

We assume that the target user has requested for recommendations over all items and sent his encrypted ratings to the server (user multiplies his rating with 100 before sending the ciphertexts of ratings). In both types of recommendations, the server generates two different ciphertexts separately [numerator and denominator of Eqs. (2) and (3)] and repeats the process for all items. Finally, the target user decrypts the results and chooses the item with highest recommendation score. The solutions to generate recommendations securely are described numerically in below.

*CBF-Based Recommendation*

For instance, we show the process of private CBF-based recommendation generation for user $u_2$ on item $i_1$. Firstly, user $u_2$ sends the message $M_2^{(7)}$ containing the ciphertexts of his ratings as follows:

$$M_2^{(7)} = \left\{E(g^0), E(g^{100}), E(g^{500}), E(g^0)\right\}$$

The server homomorphically computes the ciphertexts [numerator and denominator of Eq. (2), respectively] as

$$(A_{2,1}, B_{2,1}) = E(g^{100})^{s(i_1,i_2)} \cdot E(g^{500})^{s(i_1,i_3)} \cdot E(g^0)^{\mathrm{sim}(i_1,i_4)}$$
$$= E(g^{100})^{99} \cdot E(g^{500})^{20} \cdot E(g^0)^{98} = E(g^{19,900})$$
$$(A_{3,1}, B_{3,1}) = E\left(g^{s(i_1,i_2)+s(i_1,i_3)+s(i_1,i_4)}\right)$$
$$= E(g^{99+20+98}) = E(g^{217})$$

Server sends the message $M^{(8)}$ of these ciphertexts to target user $u_2$ by

$$M^{(8)} = \left\{(A_{2,k}, B_{2,k}), (A_{3,k}, B_{3,k})\right\}$$

User $u_2$ locally decrypts the results using his own private key $x_2$ as follows.

$$C_{4,1} = \frac{B_{2,k}}{(A_{2,k})^{x^2}} = g^{19,900}$$
$$C_{5,1} = \frac{B_{3,k}}{(A_{3,k})^{x^2}} = g^{217}$$

Using discrete logarithm user $u_2$ locally retrieve the exponent of decryption results as

$$d_1^{(6)} = \log_g g^{19,900} = 19,900$$
$$d_1^{(7)} = \log_g g^{217} = 217$$

Finally the prediction is calculated by

$$P_{2,1} = \frac{19,900}{217} = 91.71 \Rightarrow 0.91$$

Similarly, the predictions for all other items are calculated as $P_{2,2} = 0.76$, $P_{2,3} = 0.43$ and $P_{2,4} = 1.08$. Since item $i_4$ achieves highest score, it is recommended for user $u_2$. The final recommendations results are divided by 100 since the user ratings, similarities and averages were multiplied by 100 to cope with the ElGamal cryptosystem.

*CF-Based Recommendations*

Similar to the CBF, in CF-based recommendations the server generates two different ciphertexts for target user [numerator and denominator of Eq. (3)]. Let the server is generating recommendation for user $u_1$ on item $i_1$. The detailed numerical example is described as follows. Firstly, the target user $u_1$ sends his encrypted ratings as

$$M_2^{(9)} = \{E(g^{300}), E(g^{500}), E(g^0), E(g^{400})\}$$

The server computes the ciphertexts of Eq. (3)'s numerator by

$$E\left(g^{250(99+20+98)}\right)$$

and

$$(E(g^{500})/E(g^{300}))^{99} \cdot \left((E(g^0)/E(g^{350}))^{20} \cdot (E(g^{400})/E(g^{400}))^{98}\right)$$

Now the server computes the final ciphertexts of numerator (Sect. 3.2.2, step 2.1) and denominator (Sect. 3.2.2, step 2.2) homomorphically as,

$$\begin{aligned}(A_{3,1}, B_{3,1}) &= E\left(g^{250(99+20+98)}\right) \cdot \left(\left(E(g^{500})/E(g^{300})\right)\right)^{99} \\ &\quad \cdot \left(\left(E(g^0)/E(g^{350})\right)^{20} \cdot (E(g^{400})/E(g^{400}))^{98}\right) \\ &= E(g^{54,250}) \cdot E(g^{12,800}) \\ &= E(g^{67,050}) \\ (A_{4,1}, B_{4,1}) &= E(g^{99+20+98}) = E(g^{217})\end{aligned}$$

The server sends message $M^{(10)}$ to target user $u_2$ as

$$M^{(10)} = \{(A_{3,1}, B_{3,1}), (A_{4,1}, B_{4,1})\}$$

User $u_1$ receives these ciphertexts and decrypts them using his own secret key $x_1$ by

$$C_{6,3} = \frac{B_{3,1}}{(A_{3,1})^{x_1}} = g^{67,050}$$
$$C_{7,3} = \frac{B_{4,1}}{(A_{4,1})^{x_1}} = g^{217}$$

Therefore, computing discrete logarithm we find $d_1^{(8)} = \log_g g^{67,050} = 67,050$ and $d_1^{(9)} = \log_g g^{217} = 217$.

Finally, the prediction for user $u_1$ on item $i_1$ is calculated by

$$P_{1,1} = \frac{d_1^{(8)}}{d_1^{(9)}} = 308.98 \Rightarrow 3.08$$

Similarly, we get the predictions of other items for user $u_1$ as $P_{1,2} = 2.68$, $P_{1,3} = 4.48$ and $P_{1,4} = 4.67$. Since item $i_4$ achieves highest prediction score, it is finally recommended for user $u_1$.

## 4 Security Discussion

We assume our privacy-preserving recommender system protocol is based on a semi-trusted recommender server and multiple users participated in the recommendation system. The proof that our proposed solutions really fulfil the privacy requirements consist of three main observations:

1. *Security in Average Computation* To calculate averages ratings of each item, users encrypt their ratings including flags: 1 if there is any ratings, or 0 otherwise using the common public key $Y$. Thus, the ratings including which items have been actually rated are secure. All users jointly decrypt the ciphertexts of total ratings and flags [shown in Eqs. (15) and (16)] without revealing and individual's ratings. Since the server is semi-trusted, it does not collude to reveal user ratings.
2. *Security in Similarity Calculation* Users first locally compute pairwise products and square of items' ratings. Then, they encrypt these results using common public key $Y$ and send to server. Therefore, user ratings are secured. Once the server receives the ciphertexts, it homomorphically computes the similarities and allows all users to jointly decrypt the results [Eqs. (26) and (27)]. Being semi-trusted, the server does not pose any threat to user ratings.
3. *Security in Recommendations Generation*
   (a) *CBF-Based Recommendations* To generate recommendations in CBF, target user encrypts item preferences using own public key $y_i$ and sends them to the server. The server homomorphically

**Table 5** Computation and communication cost of the proposed model

| Computations | Computation cost | | Communication cost | |
|---|---|---|---|---|
| | User | Server | User | Server |
| Average | $4m\ (e)$ | $2\mathrm{m}(\mathrm{n}-1)\ (mul)$ | $6m(l)$ bits | $6mn\ (l)$ bits |
| Similarity | $3\left(\frac{m(m-1)}{2}+m\right)(e)$ | $\frac{m(m-1)}{2}(n-1)(mul)+m(n-1)(mul)$ | $3\left(\frac{m(m-1)}{2}+m\right)(l)$ bits | $3n\left(\frac{m(m-1)}{2}+m\right)(l)$ bits |
| CBF-based recommendation | $(2m+2)(e)$ | $2((m-1)(mul)+(e))$ | $2(m+2)(l)$ bits | $2(m+2)(l)$ bits |
| CF-based recommendation | $2m(e)$ | $6(e)+m(mul)$ | $2(m+2)(l)$ bits | $2(m+2)(l)$ bits |

**Table 6** Accuracy

| Computations | Computation cost | | Communication cost | |
|---|---|---|---|---|
| | User (s) | Server (s) | User (Mb) | Server (Mb) |
| Average | 0.6 | 2.14 | 0.15 | 144.84 |
| Similarity | 45.2 | 107.92 | 7.7 | $7.27 \times 10^3$ |
| CBF-based recommendation | 0.3 | 0.001 | 0.05 | 0.05 |
| CF-based recommendation | 0.3 | 0.005 | 0.05 | 0.05 |

generates ciphertexts of recommendations leveraging the item–item similarity, which is already available to it, thereby sends the ciphertexts to target user. While generating recommendations, the similarities among the items are encrypted using target user's public key $y_i$. The ciphertexts are decrypted by the user's own secret key $x_i$. Therefore, during this process target user's personal ratings and recommendations results are not revealed and thus secure.

(b) *CF-Based Recommendations* Similar to the CBF-based, CF-based process generates recommendations using ciphertexts of user's ratings and items' similarities except one additional operation: subtracting item's average from corresponding item's rating. In this case, the item's rating is already encrypted by the user and average is stored in server in plaintexts format. To overcome this situation, server encrypts the average rating using target user's public key $y_i$ and performs this subtraction homomorphically. Other operations remain same with CBF-based process. Therefore, user's ratings as well as the recommendation results are secure during CF-based recommendations generation.

## 5 Performance Evaluation

### 5.1 Theoretical Analysis

According to our proposed model, the computation and communication costs are calculated by reference to the number of items and users in the systems. Table 5 summarizes the costs to perform average computation, similarity calculation and recommendation generation by users and server, where $n$ and $m$ represent the number of users and items, respectively. We assume all users participate to calculate averages and similarities among the items in the system and only one user (target user) participates in recommendation generation. According to our method, we also assume that users encrypt their ratings and send the ciphertexts in parallel to the server, thus the computation cost on user side can be reduced by computing for one user only (shown in Table 5—average and similarity computations for user). On the server side, the computation and communication costs are represented for all users participating in the system since they depend on collaboration of all users with server. For the performance measurements we consider the time required for modular exponentiations and multiplications only which are denoted as $e$ and mul, respectively. We also assume that the communication cost is linear to the number of ciphertexts sent and received. In our model, the size of one ciphertext is considered as $l = 1024$ bits.

### 5.2 Performance Analysis

The performance analysis of our proposed model is conducted in two parts. We first analyse our method in-terms of computation and communication costs which infer the efficiency in privacy and secondly we analyse the method in-terms of recommendation accuracy. To conduct the experiment, we use Java 2 SE 8 platform with OS Windows 7, 64 bit and 3.6 GHz—core i7, 8GB CPU unit. Java cryptographic-based libraries are also used for our

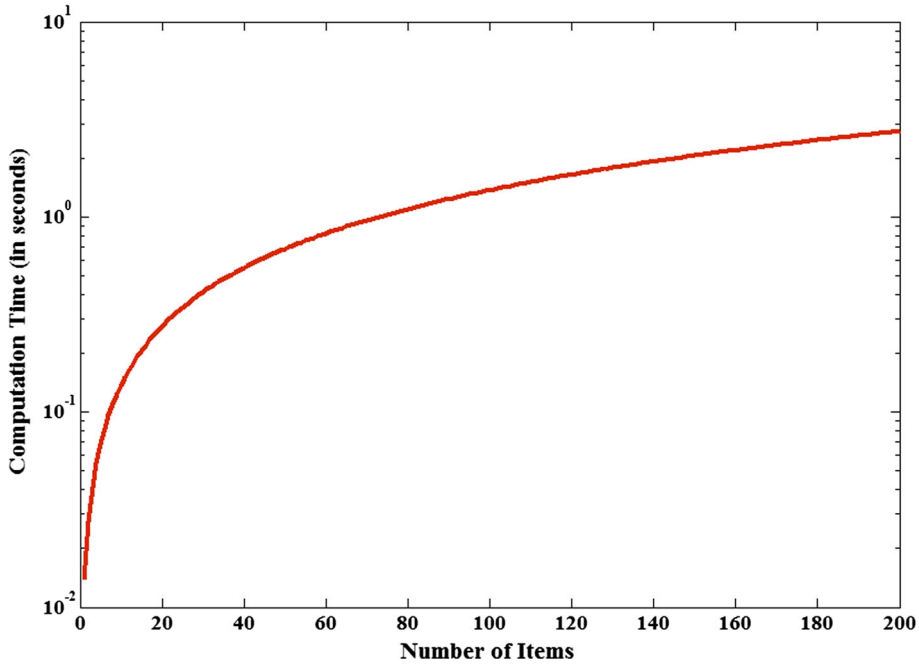


**Fig. 3** Computational cost of average calculation

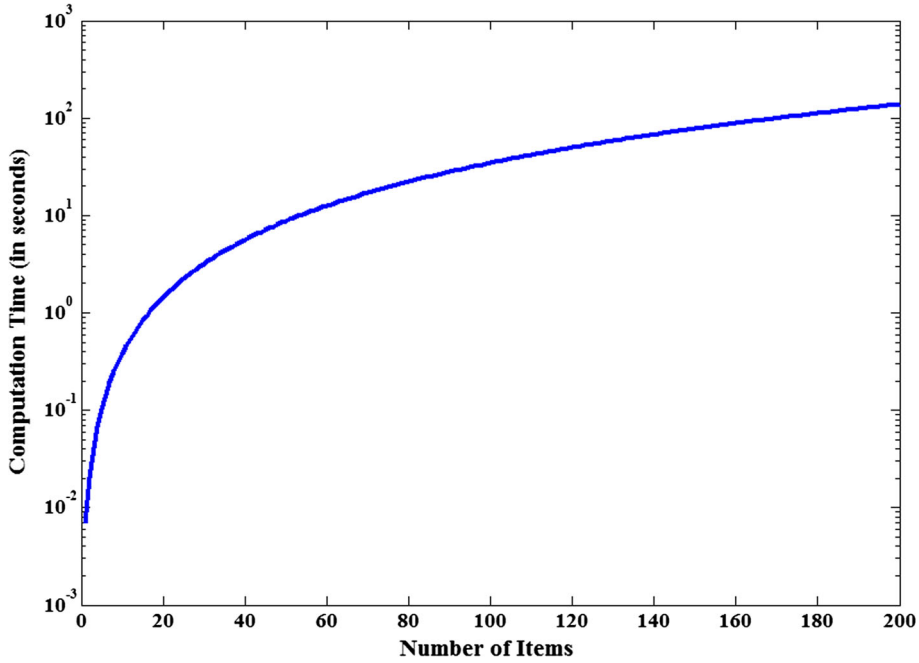


**Fig. 4** Computational cost of similarity calculation

experiment. The proposed method is evaluated using publicly available data provided by GroupLens [16] which consists of 100,000 ratings, 1982 items provided by 943 users on a scale of 1–5. In our experiments, we choose 200 items and assign 943 users who have rated on those items. Therefore, the performance analysis of our model consists of 943 users and 200 items. Once the items' averages and similarities are calculated, one user ("target user") is randomly assigned for recommendations generation.

#### *5.2.1 Computation and Communication Costs*

The implementation of our proposed method is able to compute one modular exponentiation ($e$) and multiplication ($mul$) in $7.5 \times 10^{-4}$ and $5.7 \times 10^{-6}$ s, respectively. Table 6 shows performance results (in seconds) required to perform each individual computation and amount of data that exchanged between users and the server. From this table it is clear that the performance of our proposed method is

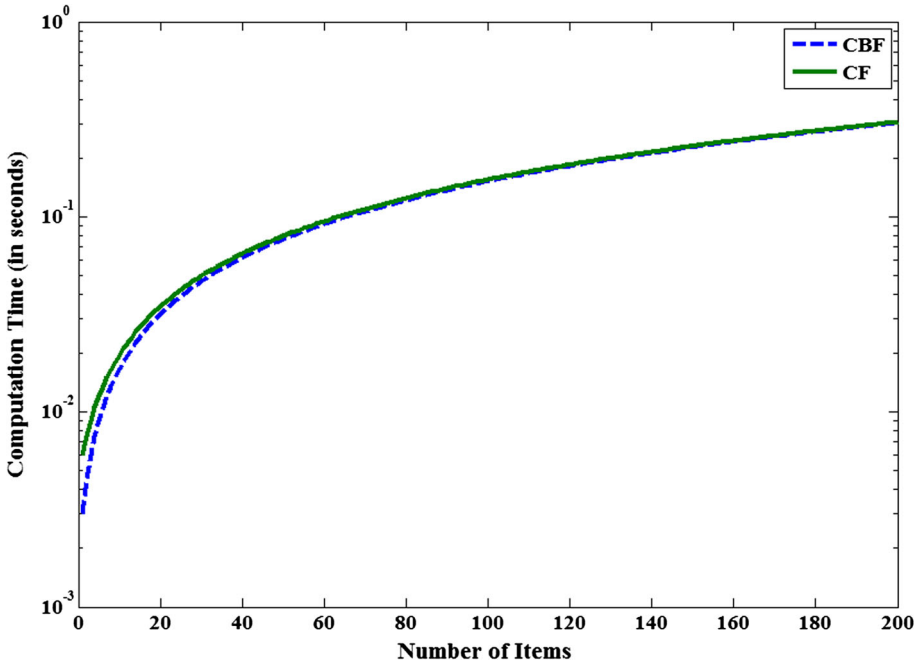


**Fig. 5** Computational cost of recommendation generation

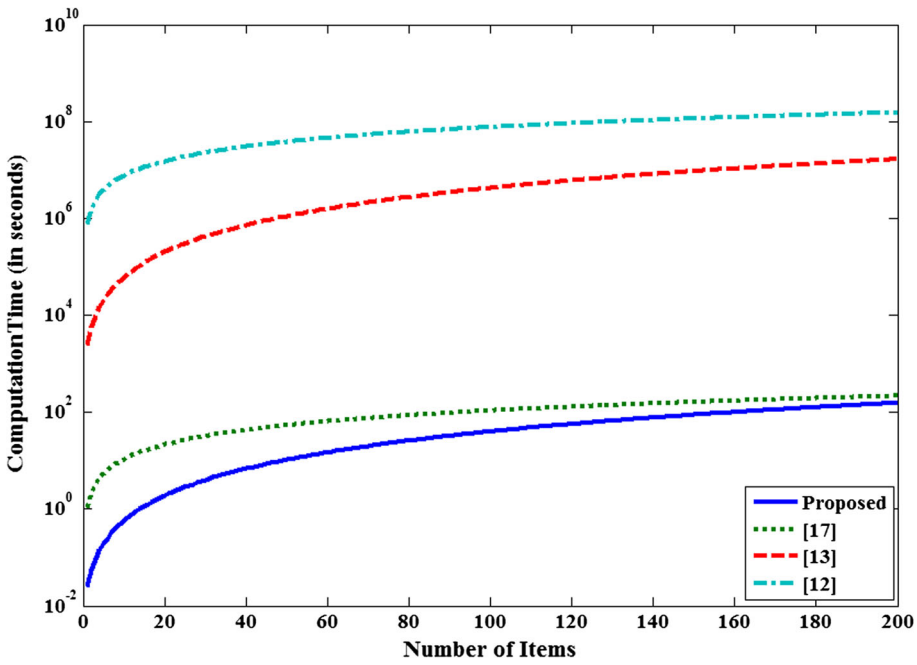


**Fig. 6** Comparison in-terms of total computation cost

efficient and cost-effective in-terms of both computation and communication.

Figure 3 shows computational time in-terms of average calculation. The results in this figure demonstrate that, our proposed privacy-preserving average computation takes only 2.7 s to calculate averages of 200 items where the number of users is 943. This confirms high efficiency in-terms of average computation. Figure 4 represents computational cost for similarity computations. Figure 5 shows the computation time in-terms of CBF and CF-based recommendations together where both methods are highly efficient and take almost equal time to generate recommendations (0.3 s for 200 items). From these analyses, we can notice that the similarity calculation takes higher time compared to other results since this part includes comparatively large computations.[5] We assume that this

[5] Note that, the server has to compute the averages and similarities among the items using all users' rating information. Since these computations depends on the number of users participated in the

computation is done only once at the beginning of recommendation process. Therefore, the computational cost for a similarity calculation does not affect the efficiency of recommendations generation.

The method is compared against three well-known privacy-preserving collaborative filtering algorithms. The first method is a randomized perturbation-based method proposed in [17], by which random noises are injected to users' ratings to prevent the recommender server from invading user privacy. Another method is a homomorphic-based solution by which authors used the Paillier cryptosystem to secure user privacy in user-based CF algorithm [12]. The last one is also based on Paillier cryptosystem by same authors where [12] was improved in-terms of computation and communication costs by introducing privacy-preserving item-based CF algorithm [13]. Figure 6 shows the comparison with [12, 17] and [13] in-terms of total computation time required to perform average, similarity calculation and recommendation generation (we consider only one type of recommendation; for instance, CF-based recommendation). Note that, Kikuchi et al. [12] and Tada et al. [13] introduced public key-based cryptographic protocol and [17] represented non-cryptographic-based solutions to generate recommendations. Therefore, from Fig. 6 we can observer that although our proposed method is based on public key-based cryptosystem, it outperforms other public key-based cryptographic solutions to a great extent. Moreover, it outperforms the non-cryptographic-based solution as well, which infers high efficiency by reducing computation overhead.

#### 5.2.2 *Recommendation Accuracy*

The proposed method does not have any accuracy loss during recommendation generation. We carried out the experiments twice with and without the security protocol to check if there is any effect in recommendation results. We found that there is no loss of recommendation accuracy while the proposed security protocol is added into recommendation process.

## 6 Conclusion

This paper presents a privacy-preserving recommender system based on item–item similarity, which can protect the user profile and rating history from any third parties or even from other users. Moreover, the server is able to compute the desired computations for recommendations without compromising the true rating information. The experimental results of our proposed model demonstrate high accuracy in-terms of the computation and communication costs as well as assuring the privacy, while the existing works suffer by failing to maintain the balance between them. Furthermore, the proposed method outperforms other cryptographic-based techniques in-terms of computational cost to generate recommendations. Our future work includes developing more efficient and secure recommender system using user-based similarity.

Footnote 5 continued
system, the required time and amount of data exchanged become higher than other computations' costs.

**Compliance with Ethical Standards**

**Conflicts of interest** The authors declare that they have no competing interests.



## References

1. Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on computer supported cooperative work. ACM, pp 175–186
2. Bilge A, Kaleli C, Yakut I, Gunes I, Polat H (2013) A survey of privacy-preserving collaborative filtering schemes. Int J Softw Eng Knowl Eng 23(08):1085–1108
3. Zhang S, Ford J, Makedon F (2006) Deriving private information from randomly perturbed ratings. In: SDM. SIAM, pp 59–69
4. Polat H, Du W (2005) SVD-based collaborative filtering with privacy. In: Proceedings of the 2005 ACM symposium on applied computing. ACM, pp 791–795
5. Shokri R, Pedarsani P, Theodorakopoulos G, Hubaux J-P (2009) Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. In: Proceedings of the third ACM conference on recommender systems. ACM, pp 157–164
6. McSherry F, Mironov I (2009) Differentially private recommender systems: building privacy into the net. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 627–636
7. Machanavajjhala A, Korolova A, Sarma AD (2011) Personalized social recommendations: accurate or private. Proc VLDB Endow 4(7):440–450
8. Canny J (2002) Collaborative filtering with privacy. In: Security and privacy. Proceedings of the 2002 IEEE symposium on. IEEE, pp 45–57
9. Hofmann T, Hartmann D (2005) Collaborative filtering with privacy via factor analysis. In: Proceedings of the 2005 ACM symposium on applied computing, pp 791–795
10. Erkin Z, Beye M, Veugen T, Lagendijk RL (2010) Privacy enhanced recommender system. In: IEEE Benelux information theory chapter
11. Erkin Z, Veugen T, Lagendijk RL (2011) Generating private recommendations in a social trust network. In: Computational

aspects of social networks (CASoN), 2011 international conference on. IEEE, pp 82–87
12. Kikuchi H, Kizawa H, Tada M (2009) Privacy-preserving collaborative filtering schemes. In: 2009 International conference on availability, reliability and security. IEEE, pp 911–916
13. Tada M, Kikuchi H, Puntheeranurak S (2010) Privacy-preserving collaborative filtering protocol based on similarity between items. In: Advanced information networking and applications (AINA), 2010 24th IEEE international conference on. IEEE, pp 573–578
14. Zhu T, Li G, Ren Y, Zhou W, Xiong P (2013) Differential privacy for neighborhood-based collaborative filtering. In: Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining. ACM, pp 752–759
15. Yi X, Paulet R, Bertino E (2014) Homomorphic encryption and applications. Springer, Berlin
16. Harper FM, Konstan JA (2015) The movielens datasets: history and context. ACM Trans Interact Intell Syst (TiiS) 5(4):19
17. Polat H, Du W (2003) Privacy-preserving collaborative filtering using randomized perturbation techniques. In: Data mining, 2003. ICDM 2003. Third IEEE international conference on. IEEE, pp 625–628