Neurocomputing 219 (2017) 440–454

Contents lists available at ScienceDirect

Neurocomputing

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

Efficient privacy-preserving content recommendation for online social communities

Dongsheng Li[a,b,d], Qin Lv[c,*], Li Shang[b,c], Ning Gu[a,d,**]

[a] School of Computer Science, Fudan University, Shanghai 201203, PR China
[b] Tongji University, Shanghai 201804, PR China
[c] University of Colorado Boulder, Boulder, CO 80309, USA
[d] Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, PR China

ARTICLE INFO

Communicated by Yanwei Pang

Keywords:
Recommendation
Privacy
Efficiency

ABSTRACT

In online social communities, many recommender systems use collaborative filtering, a method that makes recommendations based on what are liked by other users with similar interests. Privacy issues arise in this process, as sensitive personal information (e.g., content interests) may be collected and disclosed to the recommender server. Existing privacy-preserving collaborative filtering techniques trade either efficiency or accuracy for privacy, which are not suitable for online social communities with large amount of users. In this paper, we propose *YANA* (short for “you are not alone”), a user group-based privacy-preserving recommender system for users in online social communities. In this system, users are organized into groups with diverse interests and interact with the recommender server via interest-specific *pseudo users*, so that individual user's personal interest information remains hidden from the server. A suit of secure multi-party computation protocols and recommendation strategies are proposed to protect user privacy from group members in the recommendation process. A prototype system has been implemented on both mobile devices and desktop computers, and evaluation using real-world data demonstrates that YANA can effectively protect users’ privacy, while achieving high recommendation quality and energy efficiency.

1. Introduction

Online social communities have become an integral component of people's daily lives, allowing users to easily create and share content with each other, and enjoy a variety of online applications and services on a real-time basis. Therefore, recommender systems that can accurately and efficiently identify and deliver interested content to individual users have become increasingly important. Many existing recommender systems [1–8] adopt *collaborative filtering* (CF), a popular recommendation method that has high accuracy, low overhead, and is generally applicable to various application domains. CF works based on the idea that people who had similar interest in the past are likely to have similar interest in the future. In CF-based systems, the server collects user information, and then predicts a user's interest in an item based on the decisions (or ratings) of other similar users. In this process, users’ personal interests will be exposed to the recommender server, which raises privacy concerns [9,10]. Recommender servers (service providers) not only gain information regarding users’ private interests, but also, in some cases, may share users’ private data with third parties to make personalized advertisements [11]. Moreover, user privacy may be exposed to the public via the open APIs of service providers [12] or attacked by malicious users [13,14] in online communities. To address these privacy issues, a recommender system for online social communities should prevent the service provider from obtaining users’ private data while providing personalized content recommendation with high quality and high efficiency.

Recent research efforts toward privacy-preserving collaborative filtering (PPCF) have generated solutions in two main categories. One is based on secure multi-party computation (SMPC) [15–18], and the other is based on randomization [19–21]. SMPC-based methods require a large amount of computation and communication in order for users to jointly compute a value (e.g., overall rating for an item) without disclosing their personal values (e.g., individual ratings). However, in online social communities, where the number of users and content items can reach millions or even billions, SMPC-based methods are inefficient and scale poorly. Meanwhile, this kind of methods is not suitable for specific applications, such as mobile recommender systems, in which mobile devices have limited computa-

* Corresponding author.
** Corresponding author at: School of Computer Science, Fudan University, Shanghai 201203, PR China.
E-mail addresses: qin.lv@colorado.edu (Q. Lv), ninggu@fudan.edu.cn (N. Gu).

http://dx.doi.org/10.1016/j.neucom.2016.09.059
Received 19 September 2014; Received in revised form 13 March 2015; Accepted 25 September 2016
Available online 01 October 2016
0925-2312/ © 2016 Elsevier B.V. All rights reserved.

tion and communication capabilities and battery capacities. Randomization-based methods put noises on user ratings before sending them to the server. This kind of methods generally trade accuracy for privacy, which means users will receive lower-quality recommendations in order to protect their privacy. Still, randomization-based methods cannot guarantee user privacy, since the server has a very high chance to reconstruct the true ratings of the users as shown by recent works [22–24].

In this paper, we present *YANA* (short for "you are not alone"), an efficient privacy-preserving recommender system for users in online social communities. YANA can protect user privacy, while at the same time achieving high recommendation quality and energy efficiency. YANA is group based – it automatically organizes users into groups with diverse interests, so that each user's private interests can be hidden among a set of users against recommender server. A number of pseudo users are created for each user group, each pseudo user represents a unique interest, and the union of all pseudo users covers all interests of a user group. The pseudo users communicate with the recommender server on behalf of the real users. The real users can then obtain personalized recommendations based on the server's recommendations to the pseudo users and their interest distribution among pseudo users, without exposing any private data to the server. We propose four SMPC protocols for different in-group computations, which ensure user privacy from group members in the recommendation process with high efficiency.

To the best of our knowledge, this is the first complete work that tackles the efficiency problem of privacy-preserving recommender systems for online social communities.

Compared with our previous work [25], this paper improves YANA by proposing a new user group construction protocol, a novel user interest modelling method, as well as a suit of new SMPC protocols to strictly protect user privacy in user group construction, user interest modelling and content recommendation. Meanwhile, formal proofs are provided to prove the privacy preservation feature of YANA. Our key contributions are as follows:

1. We propose YANA, a novel user group-based privacy-preserving recommender system for users in online social communities, which can efficiently protect individual user's content interest privacy in the recommendation process.
2. We propose four secure multi-party computation protocols to achieve efficient privacy-preserving operations in different scenarios inside user groups. These protocols can be adopted by many applications to achieve efficient privacy-preserving computation.
3. We develop a prototype system to evaluate YANA on both mobile devices and desktop computers using real-world data. Experimental results demonstrate that, compared with state-of-the-art privacy-preserving CF approaches, YANA is much more efficient and achieves better recommendation quality.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 formulates our problem and presents a high-level overview of YANA. Sections 4–6 present the details of the YANA system design, including user grouping, pseudo user management, and the privacy-preserving recommendation algorithm. Section 7 analyzes the privacy protection capabilities of YANA. Detailed experimental results are presented in Sections 8 and 9 concludes this paper.

2. Related work

Existing works on privacy-preserving collaborative filtering (PPCF) can be classified into two main categories. Works in the first category use homomorphic encryption-based secure multi-party computation [15–18,26–30] to avoid disclosing personal information to the central server. Canny proposed a secure multi-party computation (SMPC) method on encrypted data to achieve PPCF [15], in which homomorphic encryption and distributed threshold decryption are adopted to protect user privacy while ensuring SVD-based CF. Miller et al. [16] proposed the PocketLens system, which also protects user privacy by homomorphic encryption based SMPC. Polat and Du proposed PPCF solutions on both vertically partitioned data [17] and horizontally partitioned data [18] based on homomorphic encryption. Armknecht et al. [26] proposed a distributed recommender system and an improved homomorphic encryption scheme, which can efficiently compute real numbers for recommendation. Kikuchi et al. [27] adopted homomorphic encryption to achieve three PPCF schemes: basic CF, clustering-items CF and sampling-users CF. Basu et al. [28] proposed a PPCF solution based on weighted slope one predictor, which adopts homomorphic encryption to protect user privacy. Meanwhile, they proposed how to implement such PPCF solution on the cloud in their another work [29]. Jeckmans et al. [30] proposed a PPCF solution based on homomorphic encryption, which can enable recommendations using data that are shared among different service providers. However, large-scale secure multi-party computation based on homomorphic encryption among users is inefficient and not applicable in online social communities. Our user group-based solution significantly reduces the scale of secure multi-party computation by confining the computation within each user group (with a typical group size of 10–50), thus making the system much more scalable. In addition, our solution can avoid expensive encryption and decryption operations adopted in previous works to further improve the efficiency.

Works in the second category apply randomized perturbation on users' private data before sending them to the recommender server, such that the noise can help hide user privacy [19–21]. These methods trade accuracy for privacy, which means users cannot obtain high-quality recommendations as those in non-privacy-preserving recommender systems. However, as shown by recent studies [22–24], the server may still be able to partially recover users' private information from the perturbed user data it receives via machine learning methods. McSherry et al. [31] applied differential privacy on the Netflix dataset, and they claimed that the noise introduced by differential privacy method would not significantly degrade recommendation accuracy. But still, recommendation accuracy had to be traded for privacy in their work. Recently, Shokri et al. proposed an obfuscation method to preserve user privacy [32], which obfuscates the user-item connections among similar users before user data are sent to the central server. This type of obfuscation may hide user interests in particular items, but may still reveal user interests at a higher level. For instance, the server may not know the exact items that Alice likes, but the server can infer that Alice has interest in alcoholism because most of the items after obfuscation are related to alcoholism. Meanwhile, the obfuscation among users may also hurt recommendation accuracy. Boutet et al. [33] proposed an obfuscation mechanism to hide exact profiles of users, and a randomized dissemination algorithm was proposed to ensure differential privacy during the dissemination process. Chow et al. [34] proposed a practical system for PPCF, in which users are clustered and then recommendations are generated based on the average of randomized ratings from similar users in the same cluster. Casino et al. [35] proposed a PPCF method based on micro-aggregation, in which k-anonymity can be guaranteed and better efficiency and privacy protection are achieved as compared with Gaussian noise-based PPCF method. However, as demonstrated in their studies, tradeoff between accuracy and privacy should be made in all the above three methods. Overall, randomization-based PPCF solutions are as efficient as YANA, but accuracy has to be compromised for privacy in these methods. Thus, the recommendations of these methods cannot be as accurate as YANA.

In YANA, users are organized into groups with diverse interests in order to hide each user's privacy among a group of users. This idea is similar to that of k-anonymity [36,37] and l-diversity [38] in privacy-preserving data publishing. However, with k-anonymity and l-diversity, the server anonymizes sensitive user data centrally to prevent third parties from identifying individual users or linking specific features to a user. Our problem is different and more challenging in that no central server could be trusted, and the groups should be formed and maintained in a distributed and privacy-preserving fashion.

In this work, an efficient privacy-preserving clustering algorithm is proposed to model user interest. This differs from existing works on privacy-preserving clustering [39,40] in two aspects: (a) we can optimally identify the number of user interest groups in a privacy-preserving fashion; and (b) we adopt a privacy-preserving distributed MinHash method to make clustering much more efficient.

3. Overview

In this section, we first formulate our problem and then present the high-level design of the proposed solution – YANA.

3.1. Problem formulation

In online social communities, users can post, read or comment on online posts, such as articles, pictures, music or videos via desktop computers or mobile devices. Given a user u and an online post (item) i, if u has posted/read/commented on i, we say u is interested in i, then we denote u's rating to i as $r_{i,u} = 1$. Otherwise, $r_{i,u} = 0$. Based on the binary ratings ("0" or "1"), the recommender system can generate predicted scores of items to specific users ranging in [0,1], and recommend items with high scores. Note that, we only consider the protection of user privacy on binary ratings. Other ratings, such as movie ratings on a scale of 1-5, can still be supported by YANA if these ratings are normalized into the range of [0,1], and all technique details of YANA do not need to change for such normalized ratings. For instance, we can normalize 1–5 ratings by dividing the values by 5, so that "1, 2, 3, 4, 5″ will be normalized to "0.2, 0.4, 0.6, 0.8, 1.0″, respectively. And the recommended ratings ranging in [0,1] can also be scaled back to 1–5 similarly.

Users' online activities can be accurately identified by online service providers in ordinary cases. However, with the advances of anonymous web browsing techniques [41], e.g., trusted proxy, virtual private network, etc., online users can hide their IP addresses and any other personally identifiable information from the website that they are visiting, which enables the protection of user privacy from online social communities. But recommender systems rely on user interests in the past to predict their interests in the future, so that we need a PPCF method which can protect user privacy while still providing accurate recommendations.

Among all user activities in online social communities, we denote users' post and comment information as "public" information, because users intend to interact with other users through these activities. But, users' read information does not give such intimation, and these read information may contain users' privacy. Thus, we denote users' read information as "user privacy", and to be strong enough, all users privacy (read information) should be protected from recommender server and other users during recommendation process.

3.2. System overview

To protect users' personal content interest privacy, we have proposed *YANA*, a user group-based privacy-preserving recommender system for online social communities. Using YANA, users of online social communities can obtain high-quality recommendations without sacrificing their content interest privacy to any party. Targeting at the large-scale users and items in online social communities, YANA is designed to be efficient and scalable. As illustrated in Fig. 1, YANA consists of three key components:

- **User groups**. YANA automatically organizes users into groups with diverse content interests, and individual users' content interests are hidden and aggregated within each group. Thus, user privacy is protected from the server. Inside each user group, users collaborate via privacy-preserving mechanisms, including efficient secure multi-party computation (SMPC), to protect user privacy from being inferred by other members in the same user group.
- **Pseudo users**. To obtain recommendations, the users in a user group maintain a set of pseudo users to interact with recommender server on behalf of real users. Each pseudo user represents a unique interest liked by at least one group member. The pseudo users together cover all interests of the group members. The server makes recommendations to the pseudo users based on their interests, and the real users can re-calculate their personalized recommendations based on their own interest distributions and recommendation scores to the pseudo users.
- **Recommendation algorithm**. To make recommendations, the server first needs to collect pseudo users' item ratings. We achieve item ratings through efficient secure multi-party computation inside each user group, and the aggregated item ratings of each group are sent to the server via pseudo users. The server then runs the proposed collaborative filtering algorithm to make recommendations to the pseudo users. These recommendation scores to pseudo users are used by the real users to calculate their own personalized recommendations.

4. User grouping

In this section, we describe how users are organized into user groups in a distributed and privacy-preserving fashion.

4.1. User group definition and construction

We leverage *user groups* to hide individual users' content interests among a set of users with diverse interests, such that no one can associate any specific interest with a particular user. A user group is defined as follows:

Definition 1 (*User Group*). A *user group* g is a 3-tuple: $\{\mathcal{U}_g, \mathcal{S}_g, \mathcal{P}_g\}$, in which $\mathcal{U}_g$ is a set of users who have joined g and collaborate together to hide the privacy of each other, $\mathcal{S}_g$ is the combined set of interests that users in $\mathcal{U}_g$ have, and $\mathcal{P}_g$ is a set of pseudo users who communicate with the server on behalf of the real users in g.Please note that the privacy setting of this work adopts the semi-honest model [42]. In real world

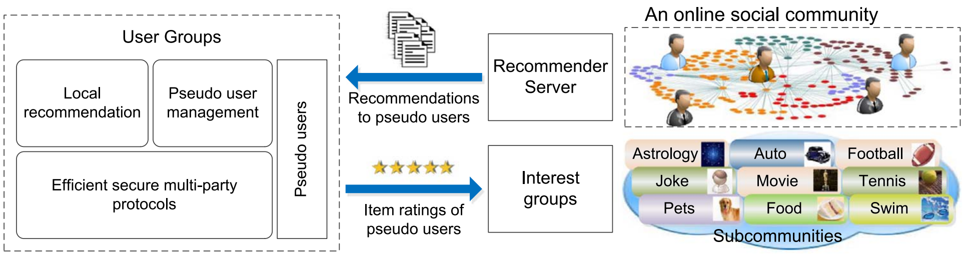


Fig. 1. YANA: efficient user group-based privacy-preserving recommendation for online social communities.

applications, totally-honest behaviors are hard to enforce, so that semi-honest behaviors are assumed in many settings [42]. In semi-honest model, users follow the computation protocols honestly except that they can store intermediate data and infer information based on intermediate data. Based on the semi-honest assumption, user groups can be formed to securely hide the privacy of each user among a set of users.

Next, we propose a group construction protocol which can automatically organize users into groups with diverse interests in a distributed and privacy-preserving fashion. Formal proofs will be given to show that user privacy can be protected inside user groups and the construction procedure would not reveal any user privacy in Section 7. The overall procedure of user group construction is described in Algorithm 1.

Algorithm 1. SecureConstruct(U).

```
Require: $U$ is the set of users in the system.
1: For each user $u \in U$, $K_u$ is the expected user group size of $u$;
2: while Not all users in $U$ are in user groups do
3:    for each $u \in U$ who has not joined any user group do
4:       $u$ chooses to be the "host" of a user group with probability $Pr_{host}(u) = K_u/|U|$;
5:       if $u$ is host then
6:          $u$ invites its friends to join its group;
7:       end if
8:    end for
9:    for each do
10:      Let $H_u$ be the set of $u$'s friends who are hosts of user groups;
11:   if $H_u \neq \varnothing$ then
12:      $u$ randomly chooses $v \in H_u$ and joins the group of $v$;
13:   else
14:      Let $J_u$ be the set of $u$'s friends who already joined user groups;
15:      if $J_u \neq \varnothing$ then
16:         $u$ randomly chooses $v \in J_u$ and joins the group of $v$;
17:      end if
18:   end if
19:   end for
20:   for each user group $g$ do
21:      Let $u$ be the host of $g$;
22:      if $|\mathcal{U}_g| \geq K_u$ then
23:         $g$ is formed;
24:      end if
25:      end for
26:   end while
```

For each user group g constructed by Algorithm 1, the number of users in g — K_g should be no less than 3. If g only contains two users (u_1 and u extsubscript2), then u_1's privacy could be easily inferred from their jointly computation results by u_2, and vice versa. For a user group g with $K_g \geq 3$, we prove that all users' privacy can be protected in Section 7. It should be noted that users may choose to leave a user group due to various reasons after the user group has been formed. Thus, once a user requests to leave, the other users in the same user group should check whether their requirements are met, i.e., whether their privacy can be protected within the user group. If all their requirements are still met, then the user group does not need to be changed. Otherwise, they should dismiss their user group, and re-construct new user group using the above **SecureConstruct** algorithm.

4.2. User interest modelling

User interest modelling can discover internal interest groups inside each subcommunity, which are more adequate in reflecting users' true interests. For instance, in the subcommunity of "News", interest groups such as "Business News", "Technology News", "Sports News" etc., could be discovered. A subcommunity may draw the attention of different sets of users, but a interest group could be only interested to specific set of users. User interest modelling is beneficial to YANA in two folds: 1) user interest are more focused in each interest group, and recommendations inside interest groups could ensure that no "noise" ratings from uninterested users are considered during recommendation process; and 2) the interest groups can also help generate pseudo users in user groups, we will show how to generate and maintain pseudo users based on the interest groups in later sections.

In YANA, we model user interest by a privacy-preserving user interest clustering algorithm, which clusters similar items into interest groups. After interest group modelling, each user will have an interest distribution over these interest groups based on the items they like and the interest groups that those items belong to. Here, k-centroids clustering method [43] is adopted to cluster similar items. As a variant of the classic k-means clustering algorithm [44], k-centroids clustering works as follows: (1) Randomly select k items as the initial k centroids. (2) Compute the distances of other items to the k centroids, and assign each item to the cluster whose centroid is closest. Then, compute the item distances inside each cluster, and choose the item with the smallest average distance to other items in the cluster as new centroid. (3) Repeat the previous step until the centroids do not change. The reason we choose k-centroids clustering is that only item distance calculations are required during clustering process, so that privacy preservation could be performed very efficiently. Other kinds of clustering techniques, such as distribution-based clustering (e.g. EM clustering), density-based clustering (e.g. DBSCAN, OPTICS) etc., requires very complex computations during clustering process. In these methods, privacy-preserving protocols with high computation overhead (e.g. homomorphic encryption based secure multi-party computation) are required to protect user privacy, which are not very efficient in online social communities with large-scale users and items.

Two main issues need to be addressed when applying the k-centroids clustering method to our problem: 1) determining the optimal number of clusters — k; and 2) efficient calculation of item distances in a distributed and privacy-preserving way.

Optimal number of clusters — k. One key challenge in k-centroids clustering is how to choose the appropriate number of clusters — k. A better number of clusters not only helps generate more accurate interest groups, but also helps generate more accurate pseudo users. Here, we adopt an idea that is similar to X-means [45]. In X-means, Bayesian Information Criterion (BIC) is adopted to score clusterings with different k values, and the k value that achieves the highest BIC score is chosen as the optimal k value. The BIC score is defined as follows [45]:

$$BIC(C) = \hat{l}(I|C) - \frac{k(d+1)}{2}\log m \tag{1}$$

where I is the dataset, C is a clustering on I, k is the number of clusters in C, d is the dimensionality of C, m is the number of items in I and $\hat{l}(I|C)$ stands for the log-likelihood of I given clustering C. A smaller k can have smaller $\hat{l}(I|C)$, and larger k will result in bigger $\frac{k(d+1)}{2}\log m$. Thus, a balanced (optimal) k can be chosen to achieve the highest BIC score.

Using the real-world dataset we have collected, we tested the item distance distribution inside each cluster with the Anderson-Darling test [46], and found that the distribution is Gaussian. Then the probability of each item i can be computed by the server as follows:

$$\hat{P}r(i) = \frac{m_i}{m} \times \frac{1}{\sqrt{2\pi}\hat{\sigma}^d}\exp\left(-\frac{1}{2\hat{\sigma}^2}D(i,\mu_i)^2\right) \tag{2}$$

where m_i is the number of items in the cluster that i belongs to, $D(i,\mu_i)$ is the distance between i and μ_i, $\hat{\sigma} = \sum_{i \in I} D(i,\mu_i)^2/(m-k)$ is the maximum likelihood estimate (MLE) for the variance of I. Therefore, the log-likelihood is:

$$\hat{l}(I|C) = \log \prod_{i \in I} \hat{P}r(i) \tag{3}$$

$$= \sum_{i \in I} \left(\log m_i - \frac{1}{2\hat{\sigma}^2} D(i, \mu_i)^2 \right) - mR \tag{4}$$

where $R = \frac{1}{2}\log(2\pi) + d\log\sigma + \log m$ is a constant.

Based on the BIC score above, the server can run the k-centroids algorithm multiple times with different k values, and choose $k^* = \text{argmax}_{k_i} BIC(C_i)$ (k_i is the number of clusters in C_i) as optimal number of clusters.

Efficient privacy-preserving item distance calculation. Another challenge in the k-centroids clustering process is to compute the distance between items efficiently without compromising user privacy. To preserve user privacy, we need a secure multi-party computation among the users to calculate the distance between any two items. Let n be the number of users in the system; note that n is large in online social communities. Thus, secure multi-party computation among n users could be rather time-consuming. To achieve high efficiency, we propose a privacy-preserving distributed MinHash method to estimate the distance between items. MinHash is an efficient method for estimating the Jaccard similarity between two sets [47]. It has the following property:

$$\mathbf{Pr}_{h \in \mathcal{H}}[h(C_i) == h(C_j)] = \frac{|C_i \cap C_j|}{|C_i \cup C_j|} \tag{5}$$

where $\mathcal{H}$ is a set of hash functions, $h \in \mathcal{H}$ can be any random permutation function on a given set, $h(C_i)$ is the index of the first “1” after randomly permuting all the elements in C_i.

In the proposed privacy-preserving distributed hashing scheme, random permutation is achieved via anonymous random walk among users. The hash values are stored in the data structure $HashVector\langle key, value\rangle$ as shown in Fig. 2. In the anonymous random walk, each user has a predefined probability of adding its information to the $HashVector$, so that no one else knows if a user's information is contained in a $HashVector$.

Users are visited in the anonymous random walk process. Once user u receives the $HashVector$, u chooses items that he/she likes but are not contained in $HashVector$ and adds them into $HashVector$ with a predefined probability ρ_u (using a Global Unique Identifier (GUID) as “key” and the list of items as “value”). By adopting the GUID as key in the $HashVector$, each user is anonymized, so that user interest can not be associated with any specific user. Meanwhile, 128-bit GUIDs are generated from random numbers containing 122 random bits, so that the total number of unique GUIDs is 2^{122}. This number is so large that the probability that different users generate the same GUID is negligible. In the anonymous random walk process, the $HashVector$ grows as more users are visited and more items are added. After all items are added in the $HashVector$, the random walk stops. Then, the final user will send $HashVector$ to the server through an anonymous communication protocol, so that the server cannot know whether the user who sends the $HashVector$ has added its information into the $HashVector$. After running the anonymous random walk process for multiple times, the server can estimate the Jaccard distance between items. The detailed procedure of the distributed hash algorithm is presented in Algorithm 2.

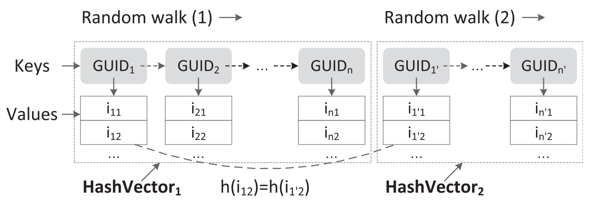


Fig. 2. HashVector in privacy-preserving distributed hashing.

Algorithm 2. SecureHash(U, I, ρ).

```
Require: U is the set of users, and I is the set of items. ρ is the
   predefined probability set of users. 1: $HashVector = \varnothing$;
1: $Hash\ Vector = \phi$;
2: The server randomly chooses user $u \in U$;
3: while Not all items in I are in HashVector do
4:     if u receives HashVector the first time then
5:         u randomly generates $0 \le r_u \le 1$;
6:         if $r_u \le \rho_u$ then
7:             u randomly generates $GUID_u$;
8:             $ItemList(u) = \{I_u - I_{HashVector}\}$;
9:             $HashVector.put(GUID_u, ItemList(u))$;
10:        end if
11:    end if
12:    u randomly selects $u' \in F_u$ ($F_u$ is the set of u's friends) and
sends HashVector to u'; (random walk)
13:    $u = u'$;
14: end while
15: while HashVector is sent to server do
16:    u sends HashVector to the server with possibility $\rho_u$; (anon-
ymous communication)
17:    if HashVector is not sent to the server then
18:        u runs one more random walk;
19:    end if
20: end while
```

Based on the **SecureHash** algorithm, given two items i_1 and i_2, their Jaccard distance can be measured by 1 minus their Jaccard similarity as follows:

$$\mathbf{JD}(i_1, i_2) = 1 - \frac{\sum \mathbb{1}(h(i_1) == h(i_2))}{\#random\ walks} \tag{6}$$

where $\mathbb{1}(x)$ is an indicator function, if $x = true$ then $\mathbb{1}(x) = 1$, otherwise $\mathbb{1}(x) = 0$.

Privacy-preserving k-centroid clustering. After calculating item distances based on the distributed hashing method, the server can cluster all the items into interest groups with different cluster numbers — k, and choose the optimal k by the BIC score based method. Then, the clustering with optimal k is the optimal interest groups clustering. These interest groups can help increase recommendation accuracy and generate pseudo users inside each user group.

4.3. Complexity analysis

4.3.1. Complexity of user interest modelling

In user interest group clustering, the server will cluster m items into k interest groups involving n users. The complexity for running **SecureHash** protocol among n users are $O(mn)$, as each user will go through all the items once visited. After the **SecureHash** protocol, the server can estimate the Jaccard Distance of items, and the complexity is $O(n*m^2)$ as the length of each $HashVector$ is $O(n)$ and $O(m^2)$ times of calculation is required to compute the distance among m items. After obtaining item distances, the server can run k-centroids algorithm to cluster m items into k clusters. As the distances are already known, the complexity of clustering is $O(k*m^2)$. At last, the BIC score is calculated for clusterings of different k values, and the complexity of which is $O(m)$. Overall, the complexity for user interest group clustering is $O(n*m^2) + O(k*m^2) + O(m)$. Although n and m could be large, the clustering computation are mainly performed on recommender server, so that polynomial complexity is acceptable.

4.3.2. Complexity of user group construction

The user groups are constructed in a peer-to-peer way. Assuming that users have constant numbers of friends and the algorithm stops within constant number of rounds, then the complexity for each user is $O(|F_u|)$, where F_u is the set of friends of user u. This is because each user only chooses one of its friends and join the friend's group in each round.

5. Pseudo user management

After user grouping, pseudo users are generated to protect user privacy during recommendation process. Real users interact with the server through the pseudo users, and all the information sent to the server by the pseudo users are the aggregated results of a group of users, so that the server cannot obtain the personal information of individual real users. Each pseudo user acts as a "delegate" for a particular interest, and the recommendations that the server makes to the pseudo user can be utilized to generate personalized recommendations by real users who have that interest.

5.1. Pseudo users generation

Pseudo users are generated based on the interests of the users inside each user group. In this section, we first present the **SecureSearch** algorithm to find the set of interests inside a given user group with privacy. In the **SecureSearch** protocol, in order to check if a set of users share a set of interests, users can run secure multi-party summation protocols in which user input is a randomly generated positive number if the user is interested in a specified interest and 0 otherwise. If the computation result is positive, it means that at least one user in the user group is interested in that interest. If the computation result is 0, it means no one in the group is interested in that interest, but does not reveal any further information. To achieve efficient secure multi-party summation, we adopt the **SecureSum** protocol [48], whose key steps are described below: (1) Each user randomly divides his/her input value into r parts such that the sum of the r parts equals to the input value. (2) Users randomly "shuffle" (send/receive) their divided parts to/from other users to obfuscate the values. (3) Each user sends the sum of his/her local obfuscated parts to the "host", and the "host" computes and returns the sum without obtaining and revealing any party's privacy. Based on the **SecureSum** protocol, the **SecureSearch** algorithm is proposed in Algorithm 3.

Algorithm 3. **SecureSearch**(g, S).

```
Require: g is a given user group. S is the set of interests.
1: $\mathcal{S}_g = \emptyset$;
2: for each interest $s \in S$ do
3:   for each $u \in \mathcal{U}_g$ do
4:   if u is interested in s then
5:       $x_{u,s} = Random(\epsilon_u, Z_u)$; ($\epsilon_u > 0$ is a small number and $Z_u \gg \epsilon_u$)
6:     else
7:       $x_{u,s} = 0$;
8:     end if
9:   end for
10:  All $u \in \mathcal{U}_g$ run a SecureSum protocol and compute $sum_s = \sum_{u \in \mathcal{U}_g} x_{u,s}$;
11:  if $sum_s > 0$ then
12:  $\mathcal{S}_g = \mathcal{S}_g \cup \{s\}$;
13:  end if
14: end for
15: return The set of interests that users in g have — $\mathcal{S}_g$.
```

Using the **SecureSearch** algorithm, users in the same user group g can find a set of interest – $\mathcal{S}_g$ liked by users in g without exposing the interest privacy of any user. Then, the group members will construct a set of pseudo users $\mathcal{P}_g$, each of which represents a unique interest in $\mathcal{S}_g$.

5.2. Pseudo users maintenance

Given a user group g, after the generation of pseudo users, users in g run a round-robin protocol to represent one pseudo user and act as the "host" when maintaining the pseudo user's interest profile. Also, to achieve load balance among users in the user group, the users choose a time interval for periodic execution of round-robin protocol to transfer pseudo user profile(s) to the successive neighbor along the "ring of users". Note that the ring of users can be formed in various ways. In YANA, we form the ring by the order of users joining g.

For each pseudo user $p \in \mathcal{P}_g$ with interest s, the interest profile of p is defined as the union of items that belong to s and are liked by users in g:

$$interest(p) = \bigcup_{u \in \mathcal{U}_g} (I_u \cap I_s) \tag{7}$$

where I_u is the set of items liked by user u and I_s is the set of items belonging to s. After the generation of pseudo users, the users in a user group should decide the item ratings of pseudo users. This step is important, as the recommender system rely on the pseudo users' decisions to make recommendations to other pseudo users (with similar interests) in other user groups. The item ratings of pseudo users could be computed as Eq. (8) in Section 6.1.

For a user group g, since no user in g would like to expose his/her interest privacy, a privacy-preserving protocol is needed to generate interest profile and item ratings for each pseudo user p. We propose the **SecureRate** algorithm to address this issue. Given pseudo user p with interest s, users in g run the **SecureSum** protocol to check if there exists users in g who are interested in items from s, and add interested items into the profile of p. Then, users in g run **SecureSum** protocol again to compute the ratings of items. The details of the **SecureRate** algorithm is presented in Algorithm 4.

Algorithm 4. **SecureRate**(g, p, s).

```
Require: g is a user group, p is the pseudo user to maintain interest s in g.
1: $s_p = \emptyset$;
2: For all $u \in \mathcal{U}_g$, u gets the set of items belonging to interest s — $I_s$ from the recommender server;
3: for each $i \in I_s$ do
4:   for each $u \in \mathcal{U}_g$ do
5:   if u likes i then
6:         $x_{u,i} = Random(\epsilon_u, Z_u)$; ($\epsilon_u > 0$ is a small number and $Z_u \gg \epsilon_u$)
7:     else
8:       $x_{u,i} = 0$;
9:     end if
10:  end for
11:  All users in $\mathcal{U}_g$ run a SecureSum protocol to compute $sum_i = \sum_{u \in \mathcal{U}_g} x_{u,i}$;
12:  if $sum_i > 0$ then
13:    $s_p = s_p \cup \{i\}$;
14:  end if
15: end for
16: for each $i \in s_p$ do
17:   Each $u \in \mathcal{U}_g$ computes $\lambda(u, s_p)$ and $r_{i,u} * \lambda(u, s_p)$ locally;
18:   All users in $\mathcal{U}_g$ run the SecureSum protocols to compute $r_{i,p}$ as Eq.(8);
19: end for
20: return: $s_p$;
```

5.3. Complexity analysis

5.3.1. Pseudo user generation

To generate a set of pseudo users inside a given user group g, users need check l interests, and one **SecureSum** protocol is required each time. As the complexity of **SecureSum** is $O(K + r)$ for each user (K is the number of users in a user group and r is the number of rounds in **SecureSum**), the total complexity is $O(l*(K + r))$ per user. As l, K and r are all not large, the cost of pseudo user generation process is small.

5.3.2. Pseudo user maintenance

During pseudo user interest profile generation, the users inside a given user group g would run the **SecureSum** protocol $O(m)$ times to determine if an item should be in the interest profile of a pseudo user, where m is the number of items. Since the complexity of **SecureSum** is $O(K + r)$ for each user, the total complexity is $O(m(K + r))$ per user. This complexity is a bit high, but acceptable as it runs only once after a user group is formed.

For pseudo user management, the users in a user group need to determine the corresponding item ratings of pseudo user. For each new item, the users need to run the **SecureSum** protocol once to calculate the goodness of the item. So the complexity of generating item ratings of pseudo users is $O(K + r)$ per user. Since K and r are not large, this cost is small for each user.

6. Privacy-preserving content recommendation

After user grouping and generation of pseudo users in each user group, the server can collect the interest profiles of pseudo users of all user groups. Then, the server can make recommendations to the pseudo users based on the item ratings of other pseudo users. However, the use of pseudo users and user groups, while protecting user interest privacy, introduces new challenges to the recommendation process:

- **Group ratings of items**. There are a set of users in each user group, some users may have shared interests. So how to measure the ratings of items within a user group is challenging.
- **Server-side recommendation**. The server cannot obtain a standard user-item rating matrix, which is required in the standard collaborative filtering algorithm. Thus, another challenge is how to make the standard collaborative filtering algorithm work in the new form of data.
- **Client-side recommendation**. In our system, the server makes recommendations only to the pseudo users, each of which represents a unique interest. But each real user may have multiple diverse interests. How to generate accurate and personalized recommendations for each real user based on the recommendations for the pseudo users is challenging.

6.1. Group ratings of new item

In this work, we leverage the interest-based collaborative filtering method, which considers the interests of individual users and the final decision is determined by users who are interested in the same type of items in the past. This guarantees that the decisions are made by the "experts", and the decisions of users who are never interested in that type of items will not be included as "negative" ratings (their weights are 0). This can help make "fair" recommendations for the items, because "noise" ratings from uninterested users are ignored in item rating.

To determine the rating of an item inside a user group, we need to consider two factors: (1) to what extent the users in the group like the item, i.e., the number of users who like the item, and (2) the importance (or "expertise") of each user in the group for this type of items. Based on these two factors, the item rating of pseudo user is computed as follows:

$$r_{i,p} = \frac{\sum_{u\in\mathcal{U}_g} r_{i,u} \times \lambda(u, s_p)}{\sum_{u\in\mathcal{U}_g} \lambda(u, s_p)} \tag{8}$$

where $r_{i,u} = 1$ if u is interested in i, and 0 otherwise. s_p is the interest profile of pseudo user p, and $\lambda(u, s_p)$ is defined as follows: $\lambda(u, s_p) = |I_{s_p} \cap I_u|/|I_u|$. In Eq. (8), weighted-average is to measure how users in the group like an item and $\lambda(u, s_p)$ is adopted as "weight" to measure how important an user is for that interest.

Given an item i and a pseudo user p inside user group g, after the item rating $r_{i,p}$ is computed by **SecureRate** algorithm, the real user who maintains p will send $r_{i,p}$ to the recommender server. Once receiving all the item ratings, the recommender server can then compute recommendation scores for pseudo users on the server side.

6.2. Server-side recommendation

For the recommender server, we adopt the memory-based collaborative filtering method, which is similar to the CF algorithm proposed by Herlocker et al. [49]. In our system, the server cannot see real user data, but only pseudo user data. Thus, the server will calculate the similarities among the pseudo users, and make recommendations to each pseudo user based on what are liked by other pseudo users with similar interests. The server generates the recommendation score of item i for pseudo user p as follows:

$$\gamma(i, p) = \frac{\sum_{p'\in P_i} r_{i,p'} \times sim(p, p')}{\sum_{p'\in P_i} sim(p, p')} \tag{9}$$

where P_i is the set of pseudo users in the system who has rated items with the same interest of i, $sim(p, p')$ is the Jaccard similarity between pseudo users p and p'. $r_{i,p'}$ is obtained from the pseudo user p'. After the calculation of $\gamma(i, p)$, the server will recommend highly-rated items to pseudo user p. It should be noted that if two pseudo users p and p' are not of the same interest then $sim(p, p')$ is 0. This means that all recommendation scores are computed among pseudo users who like that kind of items before, so that "noise" ratings from uninterested users are ignored.

6.3. Client-side recommendation

The item recommendation scores from the server can only reflect how the pseudo users like the items. After obtaining the scores, real users should re-calculate their personalized recommendation scores of items based on the server-side scores of items and their interest distributions on the pseudo users in their user groups. For instance, if a real user shares very similar interests with a pseudo user, then items recommended to the pseudo user should also be rated high to the real user. On the other hand, if a real user does not share similar interests with a pseudo user, then items recommended to the pseudo user should be rated low to the real user. Since real users may have several different interests, i.e., share different levels of interests with different pseudo users, we should combine the recommendations to different pseudo users to achieve accurate personalized recommendations to each real user. As an item may be recommended to different pseudo users with different scores, and real users also have different levels of interests for different pseudo users, we combine the scores using a linear model as follows:

$$\hat{\gamma}(i, u) = \sum_{p\in\mathcal{P}_g} \lambda(u, s_p) \times \gamma(i, p) \tag{10}$$

where $\mathcal{P}_g$ is the set of pseudo users in u's group, and s_p is the interest profile of pseudo user p. $\lambda(u, s_p)$ is adopted here to measure how important s_p is to user u, which can be computed as in Eq. (8). Based on Eq. (10), items with high $\gamma(i, p)$ will be more likely recommended to users with high $\lambda(u, s_p)$, which is reasonable as users with high $\lambda(u, s_p)$ indeed showed strong interest on s_p in the past. Please note that $\hat{\gamma}(i, u)$ is computed by each user locally, so that user privacy would not be exposed at all.

6.4. Complexity analysis

In the recommendation process, the group rating of items requires $O(|\mathcal{P}_g|)$ **SeureSum** computations, thus the complexity is $O(|\mathcal{P}_g|(K + r))$ per user. The recommendations to pseudo users are made by the server, which is $O(mn')$, where m is the number of items and n' is the number of all pseudo users in the system. For local recommendation computation, the users need to re-calculate the recommendation scores from the scores to pseudo users. The computation of this step can be done locally, and the complexity is $O(|\mathcal{P}_g|)$ for each item. Overall, the computation complexity of recommendation process is small.

7. Privacy protection in YANA

In the previous sections, the technical details of YANA have been presented. We can see that all users' privacy are aggregated among a group of users to hide the privacy of individual users. In this section, we discuss the privacy preservation feature of YANA. First, we show that the proposed user group structure can protect user privacy. Then, we prove that user privacy will not be exposed by computations inside user groups, which are user group construction, user interest modelling, pseudo user management, and content recommendation. Please note that, all the discussions are with the premise that data storages and communications are secure in the system. The case that data storages and communications are under attack is beyond the scope of this paper.

7.1. Privacy preservation feature of user groups

In YANA, the user groups (with no less than 3 users) can protect user privacy under the semi-honest model, i.e., no user is distinguishable from another user in the same user group. We prove this by the following theorem:

Theorem 7.1. *Let g be a user group constructed by Algorithm 1, and $|\mathcal{U}_g| \geq 3$. $\Pi(u_1, u_2)$ is an algorithm for user u_1 to infer the interest privacy of user u_2 in the same user group during the recommendation process (following the semi - honest behavior). Let $I_\Pi(u_1, u_2)$ denote the result of u_1 executing Π on u_2. Then, for any user $u \in \mathcal{U}_g$ and any other two users $u_1, u_2 \in \mathcal{U}_g$, $I_\Pi(u, u_1)$ and $I_\Pi(u, u_2)$ are perfectly indistinguishable.*

Proof. Let $\{j \in I | \exists\, p \in \mathcal{P}_g, r_{j,p} > 0\}$ be the set of items satisfying that $r_{j,p} > 0$ in g, where I is the set of items in the system. The only information that can be utilized by Π are u's input data $Input_u$ and the computation outputs $Output$, because 1) all the other intermediate data are random numbers in YANA, and 2) no further information could be obtained by u as users in g are not colluding in the semi-honest model.

Then, for each item $i \in \{j \in I | \exists\, p \in \mathcal{P}_g, r_{j,p} > 0\}$, the information about i is contained in $Output(i) - Input_u(i)$, where $Output(i)$ is the output of the computation on i and $Input_u(i)$ is the input of u in the computation on i. There are two scenarios to be discussed:

- $Output(i) - Input_u(i) = 0$. In this case, neither u_1 nor u_2 is interested in i, so that

  $Pr(i \in I_\Pi(u, u_1)) = 0, Pr(i \in I_\Pi(u, u_2)) = 0$

- $Output(i) - Input_u(i) > 0$. As $Output(i) - Input_u(i)$ is the combined information of u_1, u_2 and users in $\mathcal{U}_g - \{u, u_1, u_2\}$, so that $Input_{u_1}(i)$ and $Input_{u_2}(i)$ cannot be distinguished from $Output(i) - Input_u(i)$ in the semi-honest model where no users are colluding, i.e.,

  $Pr(Input_{u_1}(i) > 0) = Pr(Input_{u_2}(i) > 0)$

In both cases, we have

$Pr(i \in I_\Pi(u, u_1)) = Pr(i \in I_\Pi(u, u_2))$

Thus, we can say that $I_\Pi(u, u_1)$ and $I_\Pi(u, u_2)$ have the same distribution over $\{j \in I | \exists\, p \in \mathcal{P}_g, r_{j,p} > 0\}^*$, i.e., $I_\Pi(u, u_1)$ and $I_\Pi(u, u_2)$ are perfectly indistinguishable.□.

As any two users in the same user group are perfectly indistinguishable from the view of other parties (i.e., other users in the same group or the recommender server), so that any inferring about user privacy can only be random guesses over $\{j \in I | \exists\, p \in \mathcal{P}_g, r_{j,p} > 0\}^*$ inside user group g. And for any inferred user interest profile $\hat{I}_u \subseteq \{j \in I | \exists\, p \in \mathcal{P}_g, r_{j,p} > 0\}^*$ of user u, $Pr(\hat{I}_u = I_u) = 1/2^{|\{j \in I | \exists p \in \mathcal{P}_g, r_{j,p} > 0\}|}$ (I_u is the real interest profile of u). As $|\{j \in I | \exists\, p \in \mathcal{P}_g, r_{j,p} > 0\}|$ is usually large for user groups (i.e., over 100), so that the probability $Pr(\hat{I}_u = I_u)$ is negligible.

7.2. Privacy protection in user group construction

The user groups can hide a user's privacy among a set of users based on the above analysis. Here, we show that the user group construction process would not expose user interest privacy. The user privacy definition is adopted from Goldreich's work [42], quoted below:

Definition 2. (privacy w.r.t semi-honest behavior) [42]:

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

$$\{(S(I, (x_{i_1}, \ldots, x_{i_t}), f_I(\bar{x})), f(\bar{x}))\}_{\bar{x} \in (0,1^*)^m} \overset{def}{\equiv} \{VIEW_I^\pi(\bar{x}), OUTPUT^\pi(\bar{x})\}_{\bar{x} \in (0,1^*)^m}$$

where $OUTPUT^\pi(\bar{x})$ denotes the output sequence of all parties during the execution represented in $VIEW_I^\pi(\bar{x})$.

The above definition states that a computation protocol is privacy-preserving if the view of each party during the execution of the protocol can 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 feature of each component in YANA is the Composition Theorem in semi-honest model (Theorem 7.2). Detailed proof of Theorem 7.2 could be found in [42], and thus is omitted here.

Theorem 7.2. (*Composition Theorem for the semi - honest* model) [42]:*Suppose that g is privately reducible to f and that there exists a protocol for privately computing f. Then there exists a protocol for privately computing g.*

Based on Definition 2 and Theorem 7.2, we prove that the **SecureConstruct** algorithm does not reveal any user interest privacy.

Theorem 7.3. *Let U be a set of users in an online social community* ($|U| > 1$). *The execution of* **SecureConstruct** (Algorithm 1) *on U reveals none of the interest privacy of users in U.*

Proof. During the execution of **SecureConstruct**, the only step that contains communication is to choose a user group to join, in which no information with regards to user privacy are revealed. Therefore, the **SecureConstruct** protocol will not expose user privacy at all. □.

7.3. Privacy protection in user interest modelling

In YANA, user interests are modelled by the proposed user interest modelling algorithm. Here, we prove that the proposed user interest modelling method is privacy-preserving for users.

Theorem 7.4. *Let U be a set of users in an online social community* ($|U| > 1$)*. The execution of the proposed user interest modelling method on U is privacy - preserving for all users in U.*

Proof. During the user interest modelling process, the only step that requires communication is the distributed item similarity computation — **SecureHash** algorithm. All other steps are performed by the recommender server, during which no user interest privacy could be obtained. By applying the Composition Theorem, if we can prove that **SecureHash** is privacy-preserving, then we can prove that the user interest modelling method is privacy-preserving.

Here, we construct a simulator to simulate the three main stages of the **SecureHash** algorithm as follows:

- *Stage 1:* In this stage, a user u receives an empty *HashVector*, and decides to add its data to *HashVector* with probability ρ_u. If u chooses not to add its data, the output of u is an empty *HashVector* which could be easily simulated. Otherwise, the simulator for u can generate a random $GUID'_u$, and simulate ($GUID_u$, $ItemList(u)$) with ($GUID'_u$, $ItemList(u)$). This ensures that u's *GUID* is different each time u adds its data, such that no one can associate any user with any *GUID*. As the *GUIDs* are uniformly distributed on $(0, \ldots, 2^{122})$, the simulated output of u is indistinguishable from what the next user views in real random walk.
- *Stage 2:* In this stage, a user u receives an non-empty *HashVector*. Again, if u chooses not to add its data, the output of u could be easily simulated by shuffling the *GUIDs* in the *HashVector*. Otherwise, the simulator for u can generate a random $GUID'_u$, and simulate ($GUID_u$, $ItemList(u)$) with ($GUID'_u$, $ItemList(u)$). Then, the simulator adds ($GUID'_u$, $ItemList(u)$) into the received *HashVector*. At last, the simulator shuffles the *GUIDs* in the *HashVector*. As the shuffles would not change the order of items in *HashVector*, so that the Jaccard similarity estimation would not be affected. As *GUIDs* are uniformly distributed on $(0, \ldots, 2^{122})$, the simulated output of u is indistinguishable from what the next user views in real random walk.
- *Stage 3:* In this stage, a user u finds that all items are contained in its received *HashVector*, and u chooses to send the *HashVector* to the server with probability ρ_u. No matter what u chooses, the simulator can simulate u's output by shuffling the *GUIDs* in the *HashVector*. Then, u's output is sent to either another user or the server. Again, as *GUIDs* are uniformly distributed on $(0, \ldots, 2^{122})$, the simulated output of u is indistinguishable from what the receiver views in real random walk.

As the generation of *GUID* during simulation could be regard as constant, the above simulator is linear in the size of n (the number of users). This means that a polynomial-time simulator is successfully constructed. Thus, the **SecureHash** algorithm is privacy-preserving for all users. Then, by applying the Composition Theorem, we can say that the proposed user interest modelling method is privacy-preserving for all users in U.□.

7.4. Privacy protection in pseudo user management

In pseudo user management, the **SecureSearch** algorithm is adopted to find the set of interests inside a user group, and the **SecureRate** algorithm is adopted to compute the aggregated item rating by users inside a user group. All joint computations in **SecureSearch** and **SecureRate** are performed by the **SecureSum** algorithm inside each user group, and we have shown in previous sections that the user groups can provide secure environment for users to achieve joint computation. Here, we prove that the **SecureSearch** algorithm and the **SecureRate** algorithm are privacy-preserving in Theorems 7.5 and 7.6.

Theorem 7.5. *Let g be a user group, then the execution of* **SecureSearch** *among users in g reveals none of the interest privacy of users in g.*

Proof. We construct a simulator to simulate the stages of the **SecureSearch** algorithm as follows:

- *Stage 1:* In this stage, each user obtains the set of interests $\mathcal{S}$ from the server. This can be easily simulated by the simulator, and no user privacy is revealed during this stage.
- *Stage 2:* In this stage, all users in g check if each interest $s \in \mathcal{S}$ is interested to users in g. During this stage, the only communications among users in g are random shares of their inputs in the **SecureSum** algorithm. The simulator for each user can simulate each random share with another randomly generated number ensuring that the summation of all newly generated numbers is equal to the summation of originally shares. As all the sent/received numbers during the **SecureSum** algorithm are randomly distributed on R, so that the simulated outputs of each user are indistinguishable from the views of all other users in real computation.
- *Stage 3:* After a secure summation among users in g for interest s, if the result is greater than 0, users in g would add s into $\mathcal{S}_g$. As this result is an aggregated number randomly distributed on R, no privacy of users in g could be obtained from it. Thus, this can be easily simulated by the simulator using the returned value of the **SecureSum** algorithm.

The above simulator is linear in the size of inputs/outputs of each user, so that we can claim that a polynomial-time simulator is successfully constructed. Thus, the execution of **SecureSearch** is privacy-preserving for users in g.□.

Theorem 7.6. *Let g be a user group, then the execution of* **SecureRate** *among users in g reveals none of the interest privacy of users in g.*

Proof. The procedure of **SecureRate** is very similar to **SecureSearch**. Thus, this theorem can be similarly proved as Theorem 7.5, so formal proofs are omitted here.□.

The execution of **SecureSearch** and **SecureRate** are privacy-preserving based on Theorems 7.5 and 7.6. And all execution results of the two algorithms are aggregated information of all users in the same user group. As we have proved that any two users in the same user group are indistinguishable from the aggregated information in the semi-honest model, so we can say that the proposed pseudo user management method is privacy-preserving.

7.5. Privacy protection in content recommendation

Three steps are adopted in the proposed content recommendation process:

- 1) Item rating inside user groups. The item rating is achieved by the proposed **SecureRate** algorithm. We have discussed the privacy preservation feature of **SecureRate** in Section 7.4, which shows that user privacy can be protected against group members. And all item ratings are sent to the server via pseudo users, so that the recommender server can know nothing about the privacy of real users;
- 2) Server side recommendation. The server side recommendation are only using the interest profiles of pseudo users, so that the

privacy of real users are protected from the server. Meanwhile, we have shown in Section 7.4 that user privacy can be protected inside each user group when maintaining pseudo users. Thus, the server side recommendation is privacy-preserving;

- 3) Client side recommendation. The client side recommendations are all computed based on recommendation scores to pseudo users and user interest distribution on the interest groups on the client side. Recommendation scores to pseudo users contains no privacy of real user, and user interest distributions are stored locally on each user's machine. Thus, user privacy would not be exposed in client side recommendation.

Overall, the privacy preservation feature of the proposed content recommendation algorithm could be easily proved, so formal proofs are omitted here.

8. Experimental results

We have developed a prototype system of YANA and conducted detailed evaluation using data from *Fudan* BBS,[1] a popular online social community with mobile client support. YANA emphasizes high-quality, high-efficiency, and privacy-preserving content recommendation. In YANA, users' privacy are formally guaranteed by the privacy-preserving algorithms and protocols described in Sections 4–6. And in Section 7, we have shown that user privacy are protected in all components of YANA. Therefore, the following quantitative studies focus on recommendation quality and efficiency. Note that, all the user groups in the experiments are formed randomly.

- **Recommendation quality** is measured by *Precision* and *Recall*, defined as follows:

$$Precision = \frac{|I_u \cap I_{\hat{\gamma}}|}{|I_{\hat{\gamma}}|},\ Recall = \frac{|I_u \cap I_{\hat{\gamma}}|}{|I_u|}$$

where I_u is the set of items that user u likes and $I_{\hat{\gamma}}$ is the set of items that are recommended to u. Higher *Precision* and *Recall* indicate better recommendation quality.

- **System efficiency** is evaluated by the runtime latency, as well as energy consumption of the recommendation process on mobile devices.

We compare YANA with two state-of-the-art privacy-preserving collaborative filtering (PPCF) solutions as follows: 1) a privacy-preserving k-nearest neighbor (KNN) method proposed by McSherry and Mironov [31], which adopted the differential privacy framework to protect user privacy in the Netflix Prize Dataset. Their method outperformed the Cinematch method proposed by Netflix; and 2) a privacy-preserving SVD-based collaborative filtering framework proposed by Canny [15]. In this solution, users compute the singular value decomposition (SVD) of the user-item matrix using homomorphic encryption in a distributed way, and obtain recommendations via local computations. In KNN, the parameters are chosen as follows [31]: $\theta = 0.15$, k=20, $\beta_m = 15$, $\beta_p = 20$, and B=1. In SVD, the dimensionality is chosen as 10 [15]. Compared with the KNN and SVD solutions, YANA achieves better recommendation quality (10.4% and 14.7% improvements on average, respectively). We also compare the energy efficiency of YANA with that of SVD, as both are distributed PPCF solutions. Compared with the SVD method, YANA can reduce at least 30% on latency and 97% on energy consumption.

8.1. Experimental setup

YANA is implemented both on desktop computers and mobile devices. Desktop clients are implemented on machines with Intel Core 2 Quad 2.66 GHz CPU, 4 GB memory and 10/100 Mbps Ethernet. Mobile clients are implemented on HTC Magic smartphones. The phone runs Android 2.1 with 528 MHz CPU, 288 MB RAM, and 1340 mAh battery. Wireless communication is through Wi-Fi (54 Mbps) with approximately 220 ms latency between mobile devices. The energy consumption of YANA is measured by monitoring the runtime battery capacity of the mobile phone. The recommendation server is also developed using Java on a workstation, which collects pseudo users' information, clusters items, and computes item recommendations for the pseudo users.

Table 1
Characteristics of 12 Subcommunities Used in the Experiments.

	Astrology	Auto	Doctor	Football	Joke	KeepFit
# of users	3621	1514	1086	1641	5768	1380
# of posts	678	565	286	1064	561	211
# of views	58,726	21,413	70,100	78,012	146,933	72,249
	Movie	Music	OMTV	PIC	TV	TVEntZ
# of users	597	223	1238	7984	303	450
# of posts	119	144	195	683	265	263
# of views	4067	12,406	7528	436,612	3041	7259

YANA is tested using data from *Fudan* BBS, a popular online social community among Chinese universities. It has over 60,000 users and supports various content-related user interactions (such as posting, reading, and replying to articles, etc.) and various user social interactions (such as sending instant messages and emails, forwarding articles to friends, etc.). Everyday, there are approximately 20,000 new posts, and over 180,000 reads. *Fudan* BBS has over 100 subcommunities, and our experiments are conducted on 12 of the most popular subcommunities. The key characteristics of the 12 subcommunities are summarized in Table 1. These subcommunities vary in number of users, number of items, user-item rating matrix density, user activity pattern, user interest distribution, etc. These subcommunities allow for a comprehensive evaluation of the proposed solution, and evaluation conducted on these subcommunities could be regarded as evaluation on 12 different datasets. We have collected data over three consecutive weeks, and divided the data set into a training set (data of the first two weeks) and a testing set (data of the last week). The training set is used to construct user groups and pseudo user profiles, and the testing set is used to evaluate the quality and efficiency of recommendations.

8.2. Privacy leakage analysis of pseudo users

In existing privacy-preserving collaborative filtering methods, the server cannot obtain user privacy directly, but it can still infer user privacy by naive attack method, i.e., random guess. In YANA, user privacy may be potentially inferred if the server treats the interest of pseudo user as the interest of the real user who maintains that pseudo user. Here, we analyze the amount of privacy leaked by pseudo users in Table 2. In this study, we also adopt *Precision* and *Recall* to quantitatively measure the privacy leakage of pseudo users, and compare the leakage with that of random guess.

Table 2
Privacy Leakage of Pseudo Users.

	Attack precision	Attack recall
Random guess	5.21%	–
YANA (K=10)	4.20%	4.83%
YANA (K=20)	4.13%	4.34%
YANA (K=30)	3.99%	3.36%
YANA (K=40)	3.65%	2.39%
YANA (K=50)	3.39%	1.86%

[1] Fudan BBS, http://bbs.fudan.edu.cn.

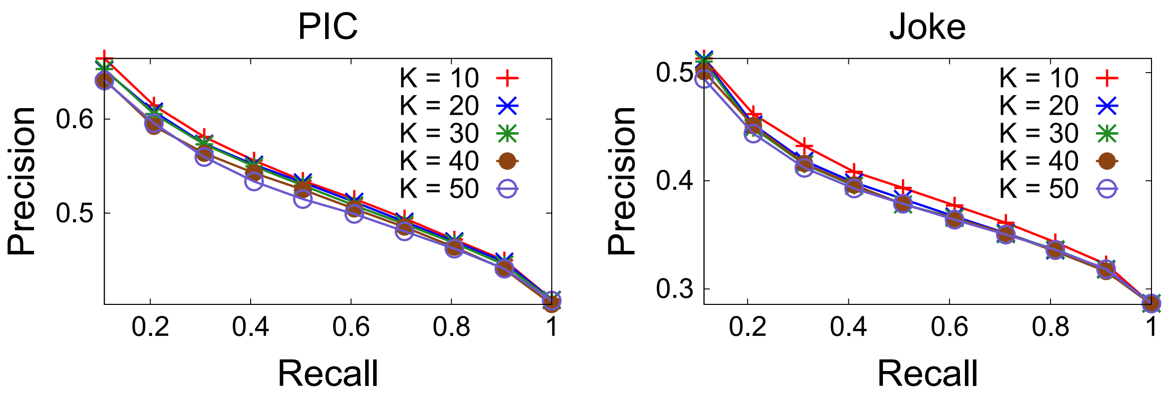


Fig. 3. Recommendation qualities with different *K* values in *PIC* and *Joke*.

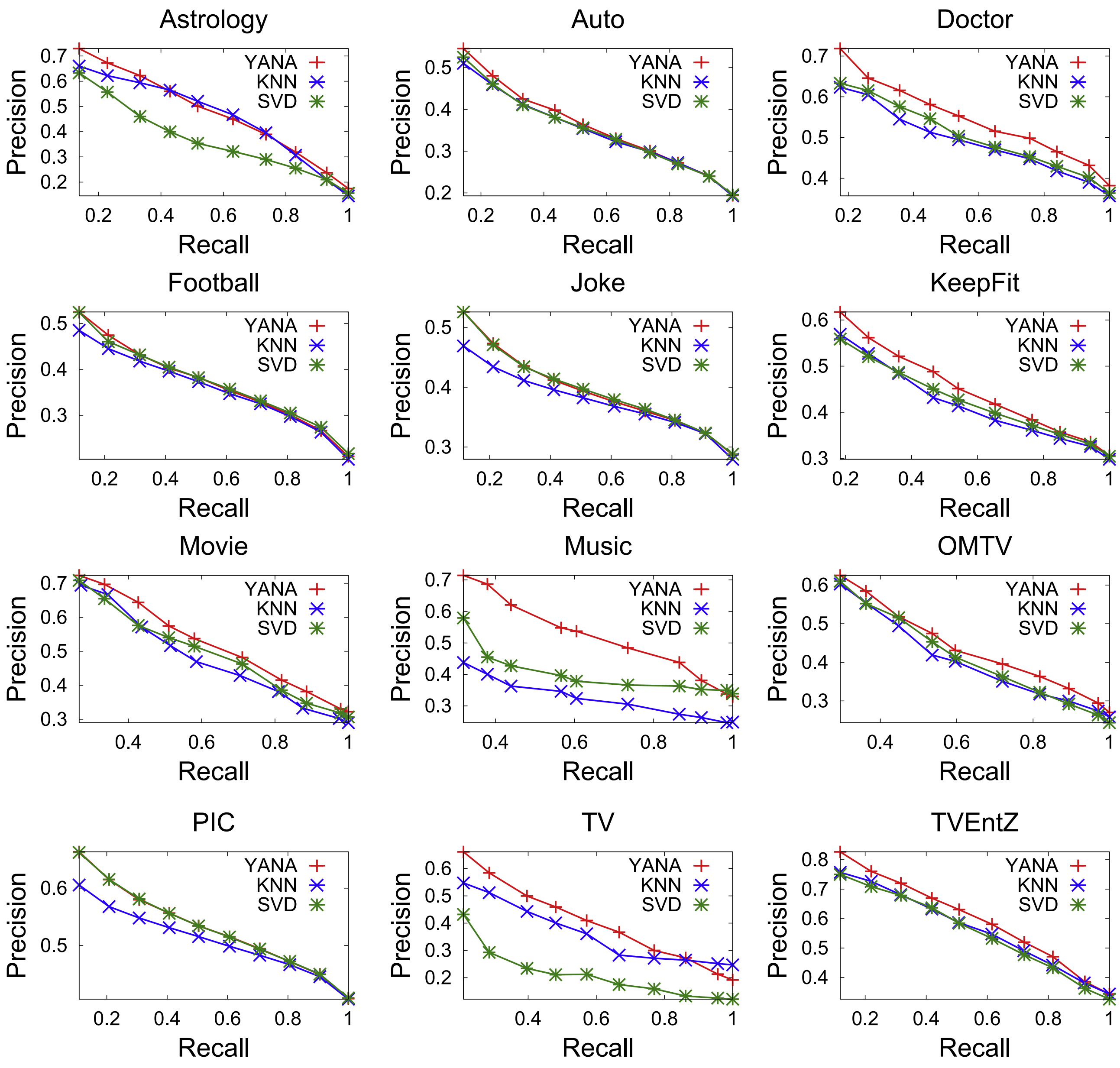


Fig. 4. Recommendation quality comparison of YANA, KNN and SVD in 12 subcommunities.

In Table 2, random guesses are performed on each subcommunity individually and the average result for all users on all subcommunities is presented. For YANA, the attack is performed by treating the interest of each pseudo user as that of the real user who maintains the pseudo user. As shown in Table 2, the attack precisions on YANA are all lower than the attack precision of random guess, which means that attacking pseudo users cannot gain more user privacy than performing random guesses. This indicates that the pseudo users can strictly protect user

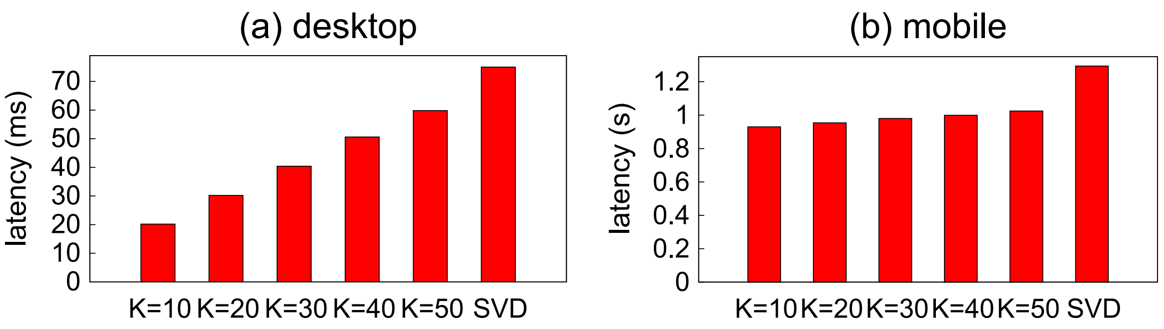


Fig. 5. Overall latency comparison of YANA and SVD of recommendation per user per item.

privacy. Meanwhile, both the attack precision and attack recall decrease as K (the size of user groups) increases. This indicates that larger user groups can provide better privacy protection.

8.3. The tradeoff between recommendation accuracy and size of user group

As described in previous sections, user group size — K may vary among different user groups. Concerns may arise that larger group size may hurt recommendation quality due to interest aggregation of more users. Fig. 3 shows the recommendation qualities with K values from 10 to 50. As we can see, recommendation precisions indeed degrade with K values increasing, but the degradations are negligible (less than 3% degradation from K=10 to K=50). Thus, we claim that YANA can achieve consistent high quality recommendation to users with different group size (i.e., different K values).

8.4. Recommendation quality

Fig. 4 shows the recommendation quality of YANA as compared with KNN and SVD algorithms.[2] As we can see, YANA outperforms KNN in all 12 subcommunities (by 10.4% on average). YANA outperforms SVD in 9 out of the 12 subcommunities (by 14.7% on average), and achieves comparable quality in the other 3 subcommunities. The improvements are achieved through the accurate interest grouping and the novel interest based recommendation, which can accurately identify user interests and ensure that item recommendations are calculated based on interested users. In three subcommunities (*Astrology*,*Music* and *TV*), YANA achieves much better quality than KNN or SVD (over 20% improvement). This is because users have much more focused content interests in these subcommunities, and identifying such focused interest groups ensures “expert opinions” while ignoring noises from non-experts. For instance, users' interests in *Astrology* are mainly focused on their own constellations. For the subcommunities where YANA has no improvements over SVD, our observation is that users have no explicit difference across different interest groups, thus do not benefit from our interest group-based recommendation. However, such scenario is not common, as users have diverse interests and different levels of interests in most subcommunities (9 out of 12). Overall, YANA achieves better recommendation quality than KNN and SVD.

8.5. Recommendation efficiency

To evaluate recommendation efficiency, we compare YANA with SVD, both of which are distributed privacy-preserving collaborative filtering algorithms. Comparisons with KNN is not considered here, as efficiencies are not directly comparable between a centralized solution (KNN) and a distributed solution (YANA). But it should be noted that the server side complexity of YANA is $O(mn)$, which is much lower than that of KNN ($O(mn^2)$). The computation and communication complexities of SVD are both $O(k'm\log n)$ per user, where k' is the decomposition factor of SVD, m is the number of items, and n is the number of users. In contrast, the computation and communication complexities of YANA are both $O(Km)$ per user. Since k' and K have similar order and n is usually large for online social communities, YANA can be much more efficient than SVD in terms of computation and communication complexities, and also energy efficiency on mobile clients. Besides comparing the latency and energy efficiency of YANA and SVD, we also measure the latency and energy consumption of basic operations of YANA to help us better understand the efficiency of YANA.

8.5.1. Latency

Fig. 5 shows the overall latency of recommending each item to a user. We only consider the overall client-side latency as it would be the bottleneck of the whole system. Please note that, for the SVD solution, we cannot measure its communication latency, as it requires a large-scale ($O(k'm\log n)$) peer-to-peer communication. Thus, we only measure the computation latency of the clients of SVD. Still, we can see that YANA outperforms SVD both on desktop computers and mobile devices in all the cases by at least 30%. The latency of YANA on desktop computers is between 20 and 60 ms, which is fairly efficient. The latency on mobile devices is around 1 s, which is also reasonable and practical for users of mobile devices.

8.5.2. Energy efficiency

We further analyze the energy consumption of YANA on mobile devices, and demonstrate that YANA is much more efficient than one of the state-of-the-art distributed PPCF solution — SVD. This study considers both the overall energy consumption and the energy consumption of basic operations.

Overall energy efficiency. Fig. 6 shows the daily energy consumption of YANA mobile clients in the twelve subcommunities, as compared with SVD. We can see that the daily energy consumption of YANA ranges from less than 0.01–0.16 mAh, which is really small compared with the total capacity of the battery (1340 mAh). In contrast, the energy consumption of SVD ranges from about 0.6–5.36 mAh, which is over 30 times higher than that of YANA. The high energy consumption of SVD is due to its encryption and decryption operations during recommendation process, which is both time consuming and energy expensive.

Energy efficiency of the SecureSum protocol. The **SecureSum** protocol, which runs in a peer-to-peer fashion is adopted to perform computations inside user groups. In a user group of size K, each user will send r messages to all other users in the group, and receive r messages from other users. r is chosen by the group members, and can be at most $K-1$ (a larger r cannot provide better privacy protection). Thus, the computation overhead is very small, as each user will only need to compute at most the sum of K numbers.

In Fig. 7, we can see that the latency increases slightly as the group size K increases, and it ranges from 400 ms to 500 ms, which is

[2] The number of users per group K is set to 10 in these experiments.

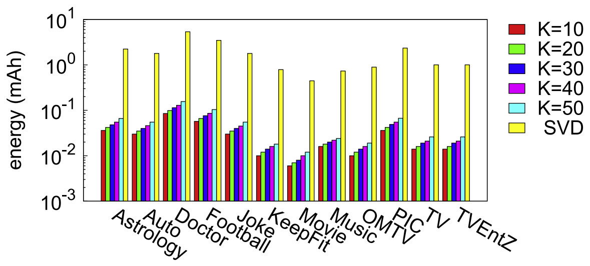


Fig. 6. Daily energy consumption comparison of YANA and SVD.

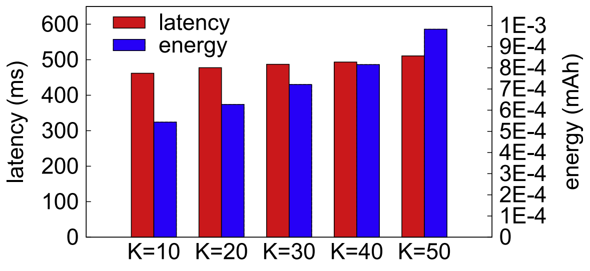


Fig. 7. Efficiency of the **Secure-Sum** protocol.

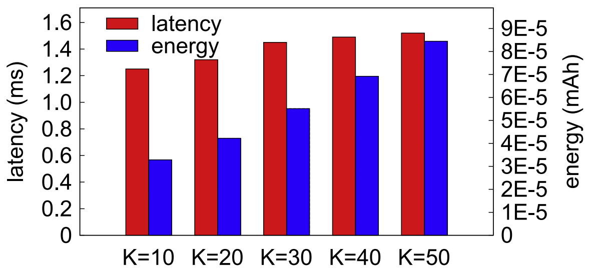


Fig. 8. Efficiency of the local recommendation calculation.

tolerable for most users (over 90% of users can tolerate a delay of 400 ms and above [50]). Energy consumption also increases with the size of user groups, and the value ranges in 0.0005–0.001 mAh, which is really small compared to the total capacity of the battery (1340 mAh).

Energy efficiency of local recommendation. The calculation of local recommendation consists of a series of local multiplications and summations, and a sorting operation. Since there is no encryption operation involved, the computation is efficient. As shown in Fig. 8, the computation time per user per item ranges from approximately 1.2 to 1.5 ms. The corresponding energy consumption ranges from 0.00003 mAh to 0.0009 mAh, which is very small compared with the capacity of the battery (1340 mAh). This study indicates that the overhead of computing local recommendation results using the recommendations to pseudo users is rather small.

9. Conclusion

The increasing popularity of online social communities raises various privacy concerns. The issue is even more severe in recommender systems of online social communities, where sensitive user content interests are collected by the recommender server in order to make personalized recommendations. In this work, we propose YANA, an efficient user group-based privacy-preserving recommender system for online social communities. YANA can organize users into user groups with diverse content interests, which can protect users' private interests from the server and other parties. Inside user groups, efficient secure multi-party computation algorithms are adopted to protect user privacy from group members. Pseudo users are created within each user group, each representing a unique interest that the group members have. The pseudo users communicate with the server on behalf of the real users in each user group, and also receive recommendations from the server. Recommendation scores can be re-calculated locally to provide customized recommendation for individual real users. We have developed a prototype system on both desktop computers and mobile smart phones, and evaluated the system using real-world data collected from an online social community. The experimental results demonstrate that YANA achieves better recommendation quality and is much more efficient compared against state-of-the-art privacy-preserving collaborative filtering solutions, while preserving user interest privacy in online social communities.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grant nos. 61233016, 61332008 and 61272533, the Open Project Program of Shanghai Key Laboratory of Data Science under Grant no. 201609060004 and the National Science Foundation of USA under Grant nos. 1251257, 1334351 and 1442971.

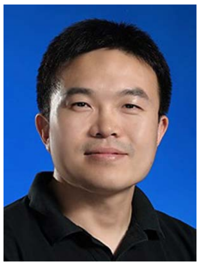
References

[1] A.S. Das, M. Datar, A. Garg, S. Rajaram, Google news personalization: scalable online collaborative filtering, in: Proceedings of the 16th International Conference on World Wide Web (WWW'07), 2007, pp. 271–280.
[2] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item collaborative filtering, IEEE Internet Comput. 7 (1) (2003) 76–80.
[3] D. Li, Q. Lv, X. Xie, L. Shang, H. Xia, T. Lu, N. Gu, Interest-based real-time content recommendation in online social communities, Knowl.-Based Syst. 28 (2012) 1–12.
[4] J. Lu, Q. Shambour, Y. Xu, Q. Lin, G. Zhang, Bizseeker: a hybrid semantic recommendation system for personalized government-to-business e-services, Internet Res. 20 (3) (2010) 342–365.
[5] J. Pan, Z. Ma, Y. Pang, Y. Yuan, Robust probabilistic tensor analysis for time-variant collaborative filtering, Neurocomputing 119 (0) (2013) 139–143.
[6] C. Chen, D. Li, Y. Zhao, Q. Lv, L. Shang, WEMAREC: accurate and scalable recommendation through weighted and ensemble matrix approximation, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2015, pp. 303–312.
[7] D. Li, C. Chen, Q. Lv, J. Yan, L. Shang, S.M. Chu, Low-rank matrix approximation with stability, in: Proceedings of the 33rd International Conference on Machine Learning, 2016, pp. 295–303.
[8] C. Chen, D. Li, Q. Lv, J. Yan, S.M. Chu, L. Shang, MPMA: mixture probabilistic matrix approximation for collaborative filtering, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence, 2016, pp. 1382–1388.
[9] F. Casino, C. Patsakis, D. Puig, A. Solanas, On privacy preserving collaborative filtering: Current trends, open problems, and new issues, in: Proceedings of the 2013 IEEE 10th InternationalConference on e-Business Engineering (ICEBE), 2013, pp. 244–249.
[10] D. Li, C. Chen, Q. Lv, L. Shang, Y. Zhao, T. Lu, N. Gu, An algorithm for efficient privacy-preserving item-based collaborative filtering, Future Gener. Comput. Syst. 55 (2016) 311–320.
[11] W. Enck, P. Gilbert, B.-G. Chun, L.P. Cox, J. Jung, P. McDaniel, A.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, 2010, pp. 1–6.
[12] G. Wondracek, T. Holz, E. Kirda, C. Kruegel, A practical attack to de-anonymize social network users, in: Proceedings of the 2010 IEEE Symposium on Security and Privacy (S & P′10), IEEE, 2010, pp. 223–238.
[13] M. Yuan, L. Chen, P.S. Yu, Personalized privacy protection in social networks, Proc. VLDB Endow. 4 (2010) 141–150.
[14] D. Li, Q. Lv, L. Shang, N. Gu, Item-based top-n recommendation resilient to aggregated information revelation, Knowl.-Based Syst. 67 (2014) 290–304.
[15] J. Canny, Collaborative filtering with privacy, in: Proceedings of the 2002 IEEE Symposium on Security and Privacy (S & P′02), 2002, pp. 45–57.
[16] B.N. Miller, J.A. Konstan, J. Riedl, Pocketlens: toward a personal recommender system, ACM Trans. Inf. Syst. 22 (2004) 437–476.
[17] H. Polat, W. Du, Privacy-preserving collaborative filtering on vertically partitioned data, in: Knowledge Discovery in Databases: PKDD 2005, 2005, pp. 651–658.
[18] H. Polat, W. Du, Privacy-preserving top-n recommendation on horizontally partitioned data, in: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), IEEE, 2005, pp. 725–731.
[19] S. Berkovsky, Y. Eytani, T. Kuflik, F. Ricci, Enhancing privacy and preserving accuracy of a distributed collaborative filtering, in: Proceedings of the 1st ACM Conference on Recommender Systems (RecSys'07), 2007, pp. 9–16.
[20] H. Polat, W. 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.
[21] S. Zhang, J. Ford, F. 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.
[22] C.C. Aggarwal, On randomization, public information and the curse of dimensionality, in: Proceedings of the IEEE 23rd International Conference on Data Engineering (ICDE'07), 2007, pp. 136–145.
[23] Z. Huang, W. Du, B. Chen, Deriving private information from randomized data, in: Proceedings of the 2005 ACM SIGMOD International Conference on Management of data (SIGMOD'05), 2005, pp. 37–48.
[24] S. Zhang, J. Ford, F. Makedon, Deriving private information from randomly perturbed ratings, in: Proceedings of the 2006 SIAM Conference on Data Mining (SDM'06), 2006, pp. 59–69.
[25] D. Li, Q. Lv, L. Shang, N. Gu, YANA: an efficient privacy-preserving recommender system for online social communities, in: Proceedings of the 20th ACM International Conference on Information and knowledge management (CIKM'11), 2011, pp. 2269–2272. (Poster).
[26] F. Armknecht, T. Strufe, An efficient distributed privacy-preserving recommendation system, in: Proceedings of the 10th IFIP Annual Mediterranean Ad Hoc NetworkingWorkshop, IEEE, 2011, pp. 65–70.
[27] H. Kikuchi, H. Kizawa, M. Tada, Privacy-preserving collaborative filtering schemes, in: Proceedings of the International Conference on Availability, Reliability and Security (ARES'09), IEEE Computer Society, 2009, pp. 911–916.
[28] A. Basu, J. Vaidya, H. Kikuchi, Efficient privacy-preserving collaborative filtering based on the weighted slope one predictor, J. Internet Serv. Inf. Secur. 1 (4) (2011) 26–46.
[29] A. Basu, J. Vaidya, H. Kikuchi, T. Dimitrakos, Privacy-preserving collaborative filtering on the cloud and practical implementation experiences, in: Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD), 2013, pp. 406–413.
[30] A. Jeckmans, Q. Tang, P. Hartel, Privacy-preserving collaborative filtering based on horizontally partitioned dataset, in: Proceedings of the International Conference on Collaboration Technologies and Systems, CTS 2012, IEEE Computer Society, 2012, pp. 439–446.
[31] F. McSherry, I. 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, New York, NY, USA, 2009, pp. 627–636.
[32] R. Shokri, P. Pedarsani, G. Theodorakopoulos, J.-P. Hubaux, Preserving privacy in collaborative filtering through distributed aggregation of offline profiles, in: Proceedings of the 3rd ACM Conference on Recommender systems (RecSys'09), 2009, pp. 157–164.
[33] A. Boutet, A.-M. Kermarrec, D. Frey, R. Guerraoui, A. Jégou, Privacy-Preserving Distributed Collaborative Filtering, Tech. Rep. RR-8253, INRIA, Mar. 2013.
[34] R. Chow, M. Pathak, C. Wang, A practical system for privacy-preserving collaborative filtering, in: Proceedings of the 2012 IEEE 12th International Conference on Data MiningWorkshops (ICDMW), 2012, pp. 547–554.
[35] F. Casino, J. Domingo-Ferrer, C. Patsakis, D. Puig, A. Solanas, A k-anonymous approach to privacy preserving collaborative filtering, J. Comput. Syst. Sci.
[36] L. Sweeney, k-anonymity: a model for protecting privacy, Int. J. Uncertain., Fuzziness Knowl.-Based Syst. 10 (5) (2002) 557–570.
[37] F. Casino, J. Domingo-Ferrer, C. Patsakis, D. Puig, A. Solanas, Privacy preserving collaborative filtering with k-anonymity through microaggregation, in: Proceedings of the 2013 IEEE 10th International Conference on e-Business Engineering (ICEBE), 2013, pp. 490–497.
[38] A. Machanavajjhala, J. Gehrke, D. Kifer, M. Venkitasubramaniam, l-diversity: Privacy beyond k-anonymity, in: Proceedings of the 22nd International Conference on Data Engineering (ICDE'06), 2006, pp. 24–35.
[39] G. Jagannathan, R.N. Wright, Privacy-preserving distributed k-means clustering over arbitrarily partitioned data, in: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD'05), 2005, pp. 593–599.
[40] J. Vaidya, C. Clifton, Privacy-preserving k-means clustering over vertically partitioned data, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'03, ACM, 2003, pp. 206–215.
[41] S. Yu, G. Zhao, W. Dou, S. James, Predicted packet padding for anonymous web browsing against traffic analysis attacks, IEEE Trans. Inf. Forensics Secur. 7 (4) (2012) 1381–1393.
[42] O. Goldreich, Secure Multi-Party Computation, Final(incomplete) Draft, Version 1.4, Oct 2002).
[43] F. Leisch, A toolbox for k-centroids cluster analysis, Comput. Stat. Data Anal. 51 (2006) 526–544.
[44] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, vol. 1, 1967, pp. 281–297.
[45] D. Pelleg, A.W. Moore, X-means: extending k-means with efficient estimation of the number of clusters, in: Proceedings of the Seventeenth International Conference on Machine Learning, ICML'00, 2000, pp. 727–734.
[46] T.W. Anderson, D.A. Darling, A test of goodness of fit, J. Am. Stat. Assoc. 49 (268) (1954) 765–769.
[47] A.Z. Broder, M. Charikar, A.M. Frieze, M. Mitzenmacher, Min-wise independent permutations, J. Comput. Syst. Sci. 60 (1998) 327–336.
[48] D. Li, Q. Lv, H. Xia, L. Shang, T. Lu, N. Gu, Pistis: a privacy-preserving content recommender system for online social communities, in: Proceedings of the 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'11), 2011, pp. 79–86.
[49] J.L. Herlocker, J.A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the 22nd Annual International ACM SIGIRconference on Research and Development in Information Retrieval (SIGIR'99), 1999, pp. 230–237.
[50] R.N. De Silva, W. Cheng, W.T. Ooi, S. Zhao, Towards understanding user tolerance to network latency and data rate in remote viewing of progressive meshes, in: Proceedings of the 20th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'10), 2010, pp. 123–128.

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.

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 modelling 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 modelling 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.

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.