Xu, Kun; Zhang, Weidong; Yan, Zheng

A privacy-preserving mobile application recommender system based on trust evaluation

Published in: JOURNAL OF COMPUTATIONAL SCIENCE

DOI: 10.1016/j.jocs.2018.04.001

Published: 01/05/2018

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:
A privacy-preserving mobile application recommender system based on trust evaluation

Kun Xu a, Weidong Zhang b, Zheng Yan a,c,∗

a The State Key Laboratory on Integrated Services Networks, School of Cyber Engineering, Xidian University, Xi’an 710071, China
b The State Key Laboratory on Integrated Services Networks, School of Telecommunications Engineering, Xidian University, Xi’an 710071, China
c Department of Communications and Networking, Aalto University, Espoo 02150, Finland

ARTICLE INFO

Article history:
Received 23 December 2017
Received in revised form 9 February 2018
Accepted 1 April 2018
Available online 3 April 2018

Keywords:
Recommender system
Mobile application
Privacy protection
Homomorphic encryption

ABSTRACT

Too many mobile applications in App stores results in information overload in App market. Mobile users are confused in choosing suitable and trustworthy mobile applications due to a large number of available candidates. A mobile application recommender system is a powerful tool that helps users solve this problem. However, there are few feasible recommender systems focusing on recommending mobile applications in the literature. First, few researches study user trust behavior based recommendation on mobile applications. Second, the accuracy and personalization of existing recommender systems need to be further improved. Particularly, privacy preservation is still an open issue in mobile application recommendation. In this paper, we propose two privacy-preserving mobile application recommendation schemes based on trust evaluation. Recommendations on mobile application are generated based on user trust behaviors of mobile application usage. In these two schemes, user private data can be preserved by applying our proposed security protocols and utilizing homomorphic encryption. We further implement two schemes and develop two mobile Apps that can be applied in different scenarios, i.e., a centralized cloud service and distributed social networking. Security analysis, performance evaluation and simulation results show that our schemes have sound security, efficiency, accuracy, and robustness.

© 2018 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

Mobile devices have become an inseparable part of people’s daily life nowadays. We are now living in a world with full support of the mobile Internet. Mobile users can do many things that are essential in their routine lives with the help of mobile applications, e.g., taking photos, enjoying entertainments, playing games, surfing web sites, taking a free riding, shopping online, social networking, reading news, consuming banking services, performing mobile payment, checking personal health, and so on. Mobile application based services have provided unbelievable convenience to mobile users, thanks to the smart phones that can access to the Internet at any time and in any places [12]. As an interface of operating smart phones, mobile applications installed in smart phones allow mobile users to enjoy various mobile Internet services and personal applications.

Mobile applications are software packages that can be executed in mobile devices [1]. To satisfy various needs of mobile users, large numbers of mobile applications are developed by manufacturers or third-party developers [2]. Fierce competitions among them result in a fact that there are so many applications with similar functions in the market, which make mobile users confused when choosing suitable applications for personal use. This is a typical phenomenon of information overload.

A recommender system is an effective tool that can be utilized to solve the problem of information overload [3,4]. A mobile recommender system is a system of generating recommendations for mobile users in a mobile Internet environment [5]. Several kinds of recommendation methods, e.g., collaborative filtering, content-based filtering and hybrid approaches [6–7], have been researched for generating recommendations in certain fields (e.g., music, movie, etc.) based on user personal profiles that mostly contain sensitive and private user information. Privacy leakage may arise without protection on data privacy and identity privacy [8,9].

1.1. Motivation

Proper selection of good applications can provide sound usage experiences to mobile users by offering high quality mobile Internet
services. User wide acceptance also impacts the success of mobile applications. However, too many mobile applications available in the mobile app market makes mobile users confused in applications choose for their personal use [10,7,11,12,14]. Some mobile applications request for excessive permissions beyond necessary and collect unnecessary information from a user, which may impact user privacy without user attention [13–16]. With the help of a recommender system, recommendations on good and credible mobile applications can be generated for users so that they can get to know applications with high quality. Although recommender systems are widely used nowadays [6,3,17], little research work has been done with regard to mobile application recommender system. A common phenomenon is that mobile users trend to download applications that have high rating scores or download numbers from mobile app stores [11,12]. But the rating score or the download number cannot ensure that an application is good enough or sufficiently suitable for a user, especially when the score or the download number cannot accurately reflect the real quality or expected quality of users. Some existing studies evaluate the quality of an application and recommend applications based on trust, or functional behaviors of applications [18,19,10,11,20,21]. But they ignore a fact that the trust behaviors of mobile users when they consume applications can greatly imply user preferences and thus offer valuable information for application recommendations. The trust behavior is a user’s actions to depend on an application or believe the application could perform as expectation, e.g., use an application regularly and continue consuming the application even facing some small problems [1]. The recommendation generated based on user trust behaviors becomes more accurate and personalized than other methods that analyze applications themselves.

However, existing mobile application recommender systems rarely take privacy preservation into concern [22,18,19,10–12,11]. User sensitive and private data are normally collected and computed when generating recommendations. In this case, privacy protection becomes important for mobile users [4,5,9,23–26]. Our previous work named TruBeRepec [1] generates mobile application recommendations based on user trust behaviors, which results in the improvement of accuracy and personalization of recommendations. Although TruBeRepec can protect user privacy to some extent by pre-processing user trust behavior data at mobile phones before sharing them with a reputation service provider, this approach cannot protect user behavior privacy with high security. A better solution is expected.

We can find some work about privacy-preserving recommendation [27,8,28–30,17,31–36]. The literature explored privacy protection and security methods from several aspects, such as trust transmission, social relationships, laws and policy, and cryptographic methods. However, existing schemes or methods cannot be directly applied into or not suitable for mobile application recommender systems due to the different context of mobile applications. How to protect user privacy while generating accurate and personalized recommendations on mobile applications based on user trust behaviors is still an open issue.

We are still facing a number of challenges to achieve a privacy-preserving mobile application recommender system. First, many private data of users are collected by the recommender system and how to control the size of data to reduce communication and processing costs is a crucial task when we design a mobile application recommender system. This issue is rarely considered in existing work. Second, how to ensure user privacy with sound security by resisting a number of attacks on the recommender system is another challenge. Finally, a successful design should be demonstrated by real implementation and performance tests. But introducing privacy protection by applying cryptographic techniques could worsen system performance, e.g., efficiency. How to make the recommender system perform efficiently in mobile devices with limited resources and achieve user acceptance is still a practical challenge.

1.2. Main contributions

In this paper, we propose two privacy-preserving mobile application recommendation schemes. The two schemes are based on our previous work named TruBeRepec [1]. We improve it by designing two privacy-preserving mobile application recommender schemes with different system structures and different operation procedures, so that they can be applied into different application scenarios. Concretely, one scheme is a privacy-preserving mobile application recommender system for cloud services, namely PPMARS-C. This scheme is a centralized recommender system that involves three kinds of system entities: mobile devices (MDs) that provide the data of user trust behaviors in terms of mobile application usage. The MD can be a smart phone for example; a cloud Service Provider (SP) that collects, stores, and processes user data for generating privacy-preserving recommendations. SP can be a cloud server with powerful storage ability and computational capability; a Privacy Center (PC) that provides necessary functionalities for data protection and user authentication. PC can be played by a certificate authority, for example. PPMARS-C is suitable for being applied in a cloud service environment.

Another scheme is a privacy-preserving mobile application recommender system in social networks, namely PPMARS-S. This scheme is a distributed recommender system that involves two kinds of system entities. One is mobile device (MD) that communicates with other devices and provides user data of trust behaviors in terms of mobile application usage. The MD can be a smart phone. The other kind of entities is a Service Provider (SP) that communicates with a certain number of MDs and generates privacy-preserving recommendations. The SP can be a base station or an edge device with sufficient computational capability. PPMARS-S is suitable for being applied into social networks in a distributed environment.

In order to improve the accuracy and personalization of recommendations, the mobile application recommendations in both schemes are generated based on the data of user trust behaviors of using mobile applications installed in their mobile devices. In order to preserve user privacy in both schemes, anonymity, public key encryption, and homomorphic encryption technology, concretely additive homomorphic encryption, and other technologies are utilized to guarantee identity security, data transmission security, and data processing security. We attempt to achieve effective, accurate, and robust recommender systems. Meanwhile, we develop two Android Apps that implement the proposed two schemes. We conduct performance evaluation by running two recommender systems and make a survey about user experiences in a real usage environment. Specifically, the contributions of this paper can be summarized as bellow:

- We propose two privacy-preserving mobile application recommendation schemes that can generate recommendations on mobile applications while preserve user privacy. The two schemes have different system structures and operation procedures so that they can be used in different application scenarios.
- We prove the security and evaluate the performance of the proposed schemes through analysis and implementation. By running the implemented recommender systems in a real context and making comparison with existing work, we show the efficiency and effectiveness of our schemes.
- We evaluate robustness of the proposed two schemes with regard to several typical attacks through simulations. The result further
shows that our schemes have sound robustness to resist internal attacks on recommendation generation.

The rest of this paper is organized as follows. Section 2 gives a brief overview on related work. System models and security models of two schemes are introduced in Section 3, followed by detailed descriptions of two schemes in Section 4. In Section 5, security analysis and performance evaluation are presented. Finally, we make a conclusion in the last section.

2. Related work

In this section, we give a brief overview on related work in terms of trust and trust behavior, privacy preservation in social networks, privacy-preserving recommender systems, and mobile application recommender systems.

2.1. Trust and trust behavior

In recent years, trust and trust behavior are getting special attention in the literature. Trust provides benchmarks in the design of security-enhanced systems, and large numbers of useful data can be obtained with regard to trust behaviors. For example, Lin and Varadarajan constructed a trust-enhanced mobile agent-based system named MobileTrust [37]. A security-centric system is shifted to a trust-centric system. Yan et al. stated that trust plays an important role in pervasive social networking and proposed a hybrid trust and reputation system for pervasive content services [38]. How to utilize user ratings and user behaviors when construct a collaborative filtering recommender system was explored in [39]. The authors discussed key features of these two kinds of data. In order to solve the drawback of traditional data management in many real-life scenarios and enhance information search on big data, the authors in [18] proposed a framework for discovering user behavioral features. The features are the result of data exchange activities when a user is interacting with intelligent systems and seeking some information (e.g., restaurant and tourist routes). Hong et al. utilized trust relations among users in context-aware recommender systems [19]. They modeled a role for a group of users with common context-aware interests and further calculated the context-aware trust value between two users. The context-aware recommendations were generated by considering both context-aware roles and trust relations. An interactive trust model for application market of the Internet Things was proposed in [21]. The trust model is constructed based on the interaction between application market and users. This model quantifies application trustworthiness by evaluating the similarity between the application’s behaviors and user’s behaviors. However, all the literature mentioned above ignore user’s trust behaviors in terms of mobile application usage, which implies user like and dislike and can provide a great deal of user data for generating accurate and personalized recommendations.

In our previous work [1], Yan et al. proposed a trust-behavior-based reputation and recommender system for mobile applications named TruBeRepec. In TruBeRepec, a model of user trust behavior for mobile applications was explored. In this model, user’s trust behaviors of mobile application usage are formalized and calculated from three dimensions: using behavior (UB), reflection behavior (RB), and correlation behavior (CB). Then, based on the calculated UB, RB, and CB of all users, a collaborative filtering method is adopted to calculate the similarities of users and further generate mobile application recommendations for users in a server. The recommendations are generated based on user real trust behaviors of mobile application usage, so the recommendations have high accuracy and are personalized. The server only collected formalized values of UB, RB, and CB from user devices, which can protect user privacy to some extent due to the pre-processing of trust behavior data. However, this approach still intrudes user privacy to some extent when processing the plaintext of statistical trust behavior data.

2.2. Privacy preservation in social networks

People nowadays have close interaction with each other due to the extensive usage of various social network software. Social network has been a hotspot of researches in the fields of data mining, trust management, reputation systems, and recommender systems [26,40,41]. Because of the complex relationships and structures in social networks, how to protect user privacy in social networks is an important task.

We can find a number of studies about privacy preservation in social networks. Based on social connections, a secure online-evaluation system was designed in [29]. This system can protect user identity while allow users to recognize evaluations from trustworthy sources. Besides, it can also preserve user relationships from an application server. However, the security of data processing was not considered. In [27], the authors explored privacy protection in a mobile ad hoc network environment and researched secure data transmission based on application and context attributes. A performance analysis model was also provided to test the proposed system. Dou et al. proposed a multimedia recommender system in the context of social networks [30]. This recommender system can preserve privacy by using a weighted noise injection technique. Concretely, core users representing the features of all users are extracted and then different noises are injected to the rating matrix of the core users. Finally, the ratings for target users are generated based on the perturbed matrix. However, this method obviously needs user rating data, which implies that the accuracy of recommendations could be seriously impacted if user ratings are not honest. Elmersy et al. proposed a multimedia service recommender system [31]. It is a privacy-aware group-based recommender system. It discovers suitable multimedia services for an interest group. There is a fog based middleware running in end-user devices, exchanging, and receiving multimedia content-related advices with others. Ometov et al. researched data security challenges in mobile social networks and pointed out that the importance of security on mobile devices was usually ignored by mobile users [42]. They implemented several security algorithms in smart phones and discovered that user sensitive data can be recovered with a great possibility if adversary utilized efficient analytical techniques. This implies the great importance and necessity to process user data securely before delivering them to networks.

2.3. Privacy-preserving recommender systems

As mentioned in Section 1, recommender systems play an important role in solving the problem of information overload by collecting a great deal of user data and generating recommendations for its users. Privacy leakage happens very easily in every procedure of the recommender system without sufficiently secure measures, such as data collection, data storage, data transmission, and data processing [43]. Privacy-preserving recommender system is get more researched in the recent literature [4]. For example, Zou and Fekri built a privacy-preserving item-based collaborative filtering recommender system by using a semi-distributed belief propagation approach [34]. Computation on similarities is transferred into a probabilistic inference problem and user rating data are protected in this system. A privacy-preserving and friendly recommender system was proposed by Guo et al. [35]. This system is based on user trust and it aims to solve the problem that people want be make friends with others while worrying about privacy
leakage when they provide social attributes such as age, name, and so on in online social networks. In this system, users apply their attributes to find matched friends and friend relationship can be established through a trust chain. What’s more, differential privacy technology was adopted in [36] to guarantee user privacy when sharing data. The authors pointed out the problem of data sparse and cold-start in a collaborative filtering system and proposed a method that leverages data from other parties to improve the richness of user data while preserve the data in a differential privacy framework. However, all of the methods mentioned above need user rating data or constructive relationships, which increase the complexity of systems.

In addition to the methods mentioned above, cryptographic methods were widely utilized to protect user private data by converting plaintext data into cipher text [44,25]. This converting makes it possible that a user’s personal data can be safely stored in a distrusted third party, instead of storing data in the user’s own mobile device with limited storage space [45]. There are many cryptographic algorithms based on different theories, and homomorphic encryption plays an important role in this area [4,46]. Homomorphic encryption allows computations over encrypted data and gets the same result as what is computed on raw data after homomorphic decryption [46]. This attribute can be applied for sensitive information protection in cloud [46].

Homomorphic encryption is divided into two categories [46]: Partially Homomorphic Encryption (PHE) and Fully Homomorphic Encryption (FHE) according to how many arithmetic operators it supports on plaintext, i.e., PHE can support either addition or multiplication while FHE supports any arithmetic operations. Uploading homomorphically encrypted user data to a third party (e.g., a distrusted cloud server with powerful storage and computational capabilities) to process can mitigate the disadvantages of processing data at mobile devices. However, because of heavy computational burden faced by FHE, PHE gets more extensive applications. Erkin et al. proposed generating recommendations by using homomorphic encryption and data packing [32]. Concretely, additive homomorphic encryption was adopted to encrypt user data. Then the encrypted data was uploaded to a semi-trusted entity named Service Provider (SP) that processes data and generates encrypted recommendations for users by running cryptographic protocols with a Privacy Service Provider (PSP). Finally, the users decrypt recommendations with the help of PSP. Besides, data packing was designed by the authors to improve efficiency. This scheme can protect user data to some extent, but there are still some drawbacks as mentioned in [32]. What’s more, malicious data provider was not considered in the proposed scheme, which makes it hard to resist internal attacks. On the other hand, identity authentication was also missed in this scheme, which may give an attacker opportunity to personate a normal user in the system, thus resulting in privacy disclosure. Homomorphic encryption was also adopted in a similar way in [33,47] for trust evaluation in some fields. However, these schemes cannot be directly applied into a privacy-preserving mobile application recommender system due to the different usage and running context of mobile applications.

There are a few mobile application recommender systems based on different methods in the literature. Mobile health applications were presented in [8,28]. The authors pointed out that insecure applications were released to App stores because of the negligence of developers, which could cause the leakage of private health information. Then, the authors classified different threat levels for applications and made recommendations based on the levels. Zhu et al. proposed a location-based mobile application recommender system in [22]. Based on the user’s real-time location, this system recommends applications for target user by considering multiple user-app factors when a user is visiting a new place. Another mobile application recommender system was proposed in [10]. The authors found that there are two main characteristics of mobile applications, one is the hierarchical structure of app markets, and the other is the competition among apps with similar functionalities. Utilizing the two characteristics, a structural user choice model was established to learn user preferences and recommendations on applications were further provided to users. He and Liu explored another way of recommending mobile applications in [111] by analyzing a psychological trait of human beings and exploratory behaviors. They proposed a goal-oriented exploratory model that integrates exploratory behavior identification with item recommendations. They also designed an algorithm for the purpose of model learning and inference in order to generate recommendations. Cao et al. [12] pointed out that existing App recommender systems largely emphasized on one single platform, so they proposed a cross-platform App recommender system. By leveraging relevant platforms of user, e.g., mobile phone and PC computers, data of the user can be enriched so that the accuracy of recommendation can be improved. Concretely, they divided user preferences on applications into two parts, i.e., applications’ inherent factors and platform-aware features, and then integrated the two parts to get final recommendations. But this system needs user numerical ratings and textual inputs from multiple platforms. Liu et al. explored the relationship between App functionality and user privacy preferences, based on which personalized mobile App recommendations are generated [20]. But they did not explore detailed privacy-preserving scheme.

In general, existing mobile application recommender systems suffer from two shortcomings. First, their application scope is limited. That is to say, existing mobile application recommender systems are limited in a certain field and one system can only recommend one kind of applications. For example, location-aware App recommender system generates recommendations on Apps related to the user’s location. Second, few of existing systems take user privacy preservation into consideration. Different ways of gathering user data are explored, but how to guarantee privacy and data security is usually ignored or missed investigation. The work presented in this paper aims to improve the above shortcomings and overcome the challenges in the research of privacy-preserving mobile application recommender systems.

3. Problem statement

In this section, we firstly introduce system models of the proposed schemes and describe the system structures of PPMARS-C and PPMARS-S, respectively. Then, we specify the security model of the two schemes in order to show the security challenges of our work.

3.1. System model

3.1.1. System structure of PPMARS-C

Fig. 1 shows the system structure of PPMARS-C. Mobile devices (MDs) (e.g., mobile phones) provide user’s personal data for appli-
cation recommendations. There is client software installed in each mobile device and the client software consists of several functional modules. Trust Behavior Monitor monitors user’s trust behaviors of mobile application usage and inputs user’s data about UB, RB, and CB into MD Database. Calculator carries out necessary computations in mobile devices, e.g., aggregation on UB, RB, and CB, encryption and decryption. Identity Manager is responsible for signature and anonymization of user identity. Key Manager generates and manages related keys. Data Disseminator sends encrypted data to Service Provider and receives data response. Recommendation Display shows the final recommendations offered by SP. All data are stored in MD Database in a safe way. SP is responsible for providing a data processing service (e.g., for the purpose of recommendation). SP, a cloud service provider for example, has powerful storage capacity and computational capacity but is curious about user privacy. There are also some modules in SP. Data Exchanger takes charge of data exchange between SP and mobile devices. SP Database stores related data. Identity Manager and Calculator conduct identity verification of mobile devices and encrypted data processing, respectively. PC is responsible for identity management of mobile devices and key management. Data Disseminator in PC is used for data communication between MD and PC. In Fig. 1, Black arrows inside each entity represent internal data flow. We assume that MDs, SP and PC communication with each other through secure channels.

3.1.2. System structure of PPMARS-S

Fig. 2 shows the system structure of PPMARS-S. Different from the structure of PPMARS-C, it is a distributed system structure that can be applied in the context of mobile social networks. Compared with PPMARS-C, the main difference of MD in PPMARS-S is that there is a Trust Value Evaluator that calculates trust values of other users in social networks. The trust value of a user is an indication that shows how trustworthy the user is. For example, the trust value can be determined according to user social networking background, knowledge, or evaluated based on user social networking behaviors. Service Provider (SP) is a server that is responsible for providing a data processing and computation service. SP communicates with a certain number of MD and generates privacy-preserving recommendations. SP can be served by a base station or an edge device that has sufficient computational capability. But SP is semi-trusted, it is curious about user private information in order to gain extra profits. There are also some functional modules in SP. A Data Exchanger takes charge of data exchange between SP and mobile devices. A SP Database stores related data. An Identity Manager conducts identity verification for mobile devices and a SP Calculator performs encrypted data processing. Involved users (i.e., MDs) constitute a group according to their social–network relationships or common attributes/interests (e.g., common hobbies). We assume that a secret is generated by trusted group members based on Shamir Threshold Protocol [48] and shared inside the group. Similarly, the black arrows in Fig. 2 inside each entity represent internal data flows and communications among system entities in the system are based on secure channels.

3.2. Difference between PPMARS-C and PPMARS-S

The difference between PPMARS-C and PPMARS-S can be described as follows. First, they hold different system structures, where SPs in different schemes have different functional modules. PPMARS-C has a centralized system structure with three types of system entities. But PPMARS-S fits into a distributed system structure that only consists of two types of system entities. Secondly, different operation procedures are designed in two schemes according to different system structures, which will be described in detail in Section 4. Finally, PPMARS-C and PPMARS-S can be applied into different application scenarios. PPMARS-C can be widely used for recommendations based on cloud services, for example, a cloud service provider offers an App recommendation service for all users in App market. PPMARS-S can work in a distributed system, such as for ad hoc group communications or social networking. For instance, a group of people who may not be friends but have common hobbies can get application recommendations from this group, which is more convenient than searching for satisfying Apps in the huge market of Apps.

3.3. Security model

The security model is common in both two schemes. It can be described as follows. First, mobile users (i.e., MDs in both schemes), SP, and PC do not collude with each other due to the concern of individual benefit and reputation. Besides, they fulfill each tasks and functionalities according to system design. Second, mobile users communicate with SP by using anonymous identities in an honest way. We assume that secure channels (e.g., SSL protocol or other encryption protocols) are utilized in the communications among system entities. Third, mobile users worry about privacy...
leakage (e.g., sensitive personal data disclosure and personal privacy mining through data analytics) when they provide data (i.e., that formalized data about UB, RB, CB, and other related data) to SP, although providing formalized data can protect user privacy to some extent. At the same time, mobile users are curious about other users’ private information (e.g., whose favorites are similar) when they request for application recommendation for himself. Forth, SP is semi-trusted in both schemes, which means that it can accomplish the functionalities according to system design, but it is curious about user privacy and may leak user privacy once it gets any possible useful information. Finally, PC is semi-trusted in PPMARS-C, it can also fulfill assigned system tasks according to system design, but it is also curious about user privacy.

4. The proposed schemes

In this section, we first give a brief introduction of homomorphic encryption, which is applied in our schemes. Then, we summarize the notations used in this paper for easy reference. Detailed procedures of the proposed two schemes are respectively described. We divide each scheme into three procedures and show how the schemes work in details.

4.1. Preliminary and notations

4.1.1. Homomorphic encryption

Due to the heavy computational complexity of fully homomorphic encryption, we utilize additive homomorphic encryption, concretely Paillier’s cryptosystem [49], to process user data. Paillier cryptosystem has two important characteristics. One is homomorphic addition of plaintexts, which means the product of two cipher texts is the cipher text of the sum of their corresponding plaintexts. We mark this characteristic as PaillierMul, as is shown in Formula (1).

$$HD(HE(m_1) \times HE(m_2)) = m_1 + m_2$$  \hspace{1cm} (1)

$HE(m_1)$ and $HE(m_2)$ are the homomorphic encryption results of plaintexts $m_1$ and $m_2$, and $HD()$ is homomorphic decryption function. As a consequence of the additive homorphism, any cipher text $HE(m)$ raised to the power $c$ results in a new encryption of $m^c$ as

$$HD(HE(m)^c) = m \times c$$  \hspace{1cm} (2)

And we mark this characteristic as PaillierExp.

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Notations</strong></td>
</tr>
<tr>
<td>$p$</td>
</tr>
<tr>
<td>$s(x)$</td>
</tr>
<tr>
<td>$s$</td>
</tr>
<tr>
<td>$N$</td>
</tr>
<tr>
<td>$l$</td>
</tr>
<tr>
<td>$T$</td>
</tr>
<tr>
<td>$ID_k$</td>
</tr>
<tr>
<td>$Sig_k(m)$</td>
</tr>
<tr>
<td>$PK_k$</td>
</tr>
<tr>
<td>$SK_k$</td>
</tr>
<tr>
<td>$E(key, m)$</td>
</tr>
<tr>
<td>$HE(key, m)$</td>
</tr>
<tr>
<td>$E_i(m)$</td>
</tr>
<tr>
<td>$T$</td>
</tr>
<tr>
<td>$T_{i,j}^k$</td>
</tr>
<tr>
<td>$T_{i,j}^k$</td>
</tr>
<tr>
<td>$T_{i,j}^k$</td>
</tr>
<tr>
<td>$Rel(u_i, u_j)$</td>
</tr>
<tr>
<td>$R_i^k$</td>
</tr>
<tr>
<td>$N_i$</td>
</tr>
</tbody>
</table>

4.1.2. Notations

For easy reference, Table 1 summarizes the notations used in this section.

4.2. PPMARS-C

In PPMARS-C, mobile users register at PC and upload preprocessed user data to SP. Then, SP calculates intermediate data for a recommendation requestor and the requestor encryps the data with homomorphic encryption and sends the data to SP. SP conducts homomorphic calculations on user data and responses...
essntial data for generating privacy-preserving recommendations. And final recommendations are calculated in a user mobile device after decrypting the data sent from SP. Specifically, PPMARS-C includes three procedures, i.e., system initialization, database construction, and interactive recommendation generation. Detailed descriptions of each procedure are presented as below.

**System initialization.** System initialization is a procedure of user registration. It has the following steps: Firstly, user k registers at PC by sending $$\text{Sig}_{\text{SP}}(ID_k)$$ to PC, where $$\text{Sig}_{\text{SP}}(ID_k)$$ is the user’s signature on the user’s anonymous identity. Secondly, PC makes a confirmation to the user’s registration and returns its signature on the user’s signature (i.e., $$\text{Sig}_{\text{SP}}(\text{Sig}_{\text{SP}}(ID_k))$$). Besides, PC generates its public key $$\text{PK}_{\text{SP}}$$ and secret key $$\text{SK}_{\text{SP}}$$, and then PC chooses a secret random number s and returns s together with $$\text{Sig}_{\text{SP}}(\text{Sig}_{\text{SP}}(ID_k))$$. Fig. 3 shows the procedure of system initialization. Note that s could be protected by applying public key encryption $$E(\text{PK}_{\text{SP}}, s)$$ when it is transmitted to user k.

**Database construction.** Mobile user’s personal data are collected from their mobile devices in this procedure. Concretely, within time period t, the Trust Behavior Monitor in a mobile device monitors the user’s trust behavior on the usage of mobile applications and collects the data related to UB, RB, and CB. Then the formalized values of UB, RB, and CB are calculated in the Calculator and stored in the Database of the mobile device. The user calculates $$E(S(t))$$ by conducting $$t = H(s)$$, where $$H(\cdot)$$ is a hash function (e.g., SHA-512) and encrypting s with $$E(S(t)) = E(\text{PK}_{\text{SP}}, s)$$, where $$\text{PK}_{\text{SP}}$$ is the public key of PC. Then, the formalized values of UB, RB, and CB are multiplied by $$E(S(t))$$ and with the user’s authorization, the multiplied results are sent to SP together with the user’s anonymous identity and timestamp t by the MD Data Disseminator. The SP receives these data and stores them in SP Database.

In the next time period t + 1, the database of the user mobile device will be updated as follows: Firstly, all users conduct $$s_{t+1} = H(s_1)$$, and $$E(S_{t+1}) = E(\text{PK}_{\text{SP}}, s_{t+1})$$. Then $$E(S_{t+1})$$ will multiply the formalized values of UB, RB, and CB. Then, the multiplied results are sent to SP together with the user’s anonymous identity and timestamp t + 1 by the Data Disseminator. SP receives these data and stores them in SP Database. Later database update is the same as what has been described above. Fig. 4 shows the procedure of database construction and update. $$T^k_i(t)_{\text{UB}}$$ represents the formalized value of user k’s UB with regard to application i at time period t, and other similar symbols follow the same style of representation.

**Interactive recommendation generation.** This is an interactive procedure including the recommendation requestor, SP, and PC. A recommendation requestor sends a request to SP firstly, and a set of protected user correlations are sent back to the requestor by PC if user validity check is positive. Then, the requestor processes the user correlations with homomorphic encryption and sends the processing result to SP. SP conducts homomorphic calculations on the processing result that are essential for generating privacy-preserving recommendations. Based on the calculation results sent from SP, the requestor generates final recommendations in his mobile device.

In this procedure, homomorphically encrypted user data are processed for providing final recommendations. Concretely, a recommendation requestor, e.g., user k sends his request ($$\text{Sig}_{\text{SP}}(ID_k), \text{Sig}_{\text{SP}}(\text{Sig}_{\text{SP}}(ID_k))$$) to SP, where $$\text{Sig}_{\text{SP}}(ID_k)$$ is k’s signature on his anonymous identity and $$\text{Sig}_{\text{SP}}(\text{Sig}_{\text{SP}}(ID_k))$$ is PC signature on k’s signature. SP checks k’s validity with the help of PC. If the check is positive, SP calculates the user k’s correlations with other users (user j) $$\text{Rel}_k(u_j, u_k)$$ based on Formula (3), otherwise, SP does nothing.

$$\text{Rel}_k(u_j, u_k) = E(S(t)) \text{ } \times \text{ } \sum_{i \neq j} \left( \sqrt{\left(T^k_i(t)_{\text{UB}} - T^j_i(t)_{\text{UB}}\right)^2 + \left(T^k_i(t)_{\text{RB}} - T^j_i(t)_{\text{RB}}\right)^2 + \left(T^k_i(t)_{\text{CB}} - T^j_i(t)_{\text{CB}}\right)^2} \right) \text{ (3)}$$

where $$T^k_i(t)_{\text{UB}}$$ represents user k’s formalized value of UB with regard to application i at time period t, and other symbols follow the same style of representation. The correlations are protected by $$E(S(t))$$ at time period t (the concrete t of E(S(t)) is determined according to which time period it is). Then, SP encrypts $$\text{Rel}_k(u_j, u_k)$$ as $$E(\text{Rel}_k(u_j, u_k)) = E(\text{PK}_{\text{SP}}, \text{Rel}_k(u_j, u_k))$$ with k’s public key $$\text{PK}_{\text{SP}}$$ and returns a set of encrypted user’s correlations $$(t, E(\text{Rel}_k(u_j, u_k)), j \neq k)$$ to user k. After getting $$(t, E(\text{Rel}_k(u_j, u_k)), j \neq k)$$, user k decrypts $$E(\text{Rel}_k(u_j, u_k))$$ with his secret key $$\text{SK}_{\text{SP}}$$. Then, it chooses $$E(S(t))$$ according to timestamp t and gets a set of user’s real correlations $$(\text{Rel}_k(u_j, u_k), j \neq k)$$ by removing $$E(S(t))$$. Note that user k cannot know the real identity of any one correlated user, user j for example, because any one user’s identity is anonymous. Then user k encrypts $$\text{Rel}_k(u_j, u_k)$$ into $$\text{HE}(\text{PK}_{\text{SP}}, \text{Rel}_k(u_j, u_k))$$ with his homomorphic encryption key $$\text{PK}_{\text{SP}}$$ and sends a set of encrypted values $$(\text{HE}(\text{PK}_{\text{SP}}, \text{Rel}_k(u_j, u_k)), j \neq k)$$ to SP. Sequentially, based on homomorphic encryption computations, SP calculates the sum of encrypted user correlations $$(\text{HE}(\text{PK}_{\text{SP}}, \sum_k \text{Rel}_k(u_j, u_k)))$$ by using homomorphic operation PaillierExp, refer to Formula (4)

$$\text{SP further calculates the sum of the encrypted aggregated result by using PaillierExp again (i.e., Formula (6))}$$

$$\text{HE} \left( \sum_{j \neq k} \text{HE}(\text{PK}_{\text{SP}}, \text{Rel}(u_j, u_k)) \right) = \prod_{j \neq k} \text{HE}(\text{PK}_{\text{SP}}, \text{Rel}(u_j, u_k)) \text{ (4)}$$

Then, SP aggregates the formalized values of UB, RB, and CB with user’s correlations by using homomorphic operation PaillierExp, refer to Formula (5).

$$\text{HE} \left( \sum_{j \neq k} \text{HE}(\text{PK}_{\text{SP}}, \text{Rel}(u_j, u_k)) \right) = \prod_{j \neq k} \text{HE}(\text{PK}_{\text{SP}}, \text{Rel}(u_j, u_k)) = \prod_{j \neq k} \text{HE}(\text{PK}_{\text{SP}}, \text{Rel}(u_j, u_k)) \text{ (6)}$$
Fig. 3. A procedure of system initialization in PPMARS-C.

Fig. 4. A procedure of database construction and update in PPMARS-C.
Algorithm 1. Homomorphic operations on encrypted user correlations

Input: \(\{HE(HPK_p, \text{Req}(u_i, u_j)); j \neq k\}\): the set of homomorphic encrypted user correlations between user k and other users;
\[
\begin{align*}
E(S_k) & = \left\{ \begin{array}{l}
\frac{1}{T_{ik}} \chi_{T_{ik}}(x), j \neq k \\
\frac{1}{T_{ik}} \chi_{T_{ik}}(x), j \neq k
\end{array} \right. \\
& = \text{the set of other users' formalized trust behavior vectors of application i at time period t, protected by } E(S_k).
\end{align*}
\]

Output: \(X; Y\): the data to calculate final recommendations.

1: for each user \(j \neq k\) do
2: for each mobile application \(i\) consumed by user \(k\) and user \(j\) do
3: Calculate \(X = \text{the result of Formula (4)}\);
4: Calculate the result of Formula (5);
5: Calculate \(Y = \text{the result of Formula (6)}\);
6: end for
7: end for
8: return \((X; Y)\)

Besides, SP also calculates parameter \(N_k\) and sends \(N_k\) to user \(k\) together with the output of Algorithm 1 and timestamp \(t\). Finally, user \(k\) decrypts the results of homomorphic operations with his decryption key \(HK_k\) and calculates the final recommendation of application \(i\), i.e., \(R^k_i\) in his mobile device based on Formula (7)

\[
R^k_i = \frac{Q}{E(S_k)} \ast N_k \quad (i = 1, 2, \ldots, l)
\]

where \(P\) and \(Q\) are the decryption results of \(X\) and \(Y\) in Algorithm 1 respectively, and \(j\) is the number of recommended mobile applications. The final recommendations on mobile applications are displayed by the Recommendation Displayer in the mobile device. Fig. 5 shows the procedure of interactive recommendation generation. Note that we exploit the check of user validity in PC for concise presentation.

It worth pointing out that we apply a timestamp in database construction and interactive recommendation generation for the purpose of security. Within each timestamp, the secret random number generated by PC is processed at the user's device in order to generate different secret numbers at different time slots to protect user data. Using anonymous identity, our scheme can protect user identity to some extent. Besides, access control can be performed by PC. What's more, user's real relationship in terms of similarity can be well protected from SP by introducing a secret number multiplication on UB, RB, and CB. And the homomorphic encryption on the user data can prevent SP from knowing exact recommendations for a user, because the final recommendations are calculated in the user’s own device, thus protecting his privacy.

In PPMARS-C, user data are stored in SP database for calculating application recommendations in a centralized way. The destruction of SP database cannot disclose any user data since it is securely protected by \(E(S_k)\). On the other hand, user data are collected automatically and periodically from user mobile device. That is to say, user database can be re-constructed even though the database is destroyed and recommendations can be generated based on the latest or recent content of database. Notable, PPMARS-S was designed to be applied into a decentralized system, which overcomes the shortcomings of PPMARS-C.

4.3. PPMARS-S

Different from PPMARS-C that is suitable for a cloud service environment, PPMARS-S is designed for a distributed social networking environment. PPMARS-S also includes three procedures, i.e., system initialization, database construction, and interactive recommendation generation. However, the details in each procedure are different from PPMARS-C, especially in the procedures of system initialization and database construction. Mobile users form a group in system initialization and a secret is generated by a most trusted user and shared among all users in this group based on Shamir Threshold Protocol [48]. The secret is generated within the group only if a certain number of users provide right sub-secrets. Database construction is initiated when a legal user sends a recommendation request to SP, and user data that are protected by the secret generated in the group are then uploaded to SP for generating privacy-preserving recommendations. Interactive recommendation generation is conducted between the requestor and SP to generate final recommendations, which is similar to what is described in PPMARS-C. Details of these procedures are described below.

System initialization. Different from the system initialization of PPMARS-C, after key generation, users constitute one group according to their social–network relationships or common attributes/interests (e.g., common hobbies), which is quite usual in real life. For example, a user would like to get recommendations from his friends can constitute a social group with his friends. Another example is a social group could be established when the users who have common interests in some applications want to obtain recommendations from others. A secret is generated based on Shamir Threshold Protocol [48] that is a \((T, L)\) threshold scheme, where \(L\) is the number of divided child secrets and \(T\) is the value of threshold.

Concretely, the most trusted user \(u\) (can be determined according to user social networking background, knowledge, or evaluated based on user social networking behaviors) in the group is selected to choose a big prime number \(p\) and construct an integral coefficient polynomial \(s(x) = a_T x_T^{T-1} + a_{T-1} x_T^{T-2} + \ldots + a_1 x + a_0 (mod p)\) over a finite field \(GF(p)\), and \(\{a_0, a_1, \ldots, a_{T-1}\}\) are random numbers. In this case, the secret \(s = a_0\). Each user \(k\) \((k=1, 2, \ldots L-1, L)\) is the number of users in the group) sends his anonymous identity \(ID_k\) to user \(u\). \(u\) calculates \(s(ID_k)\) based on the polynomial \(s(x)\) and returns \(s(ID_k)\) secretly to user \(k\). User \(u\) also calculates \(s(ID_k)\) based on \(s(x)\). Finally, any \(L\) \((L \geq T)\) users who provide \(\{ID_k, s(ID_k)\}\) will get secret \(s\) uniquely so that \(s\) can be shared within these \(L\) users. Otherwise \((i.e., L < T)\), they can never get \(s\), which means that \(s\) can be gained if and only if there are \(L\) \((L \geq T)\) users who provide correct \(\{ID_k, s(ID_k)\}\). This provides security guarantee on user validity. Note that each user within these \(L\) users will receive \((L-1)\) messages of \(\{ID_k, s(ID_k)\}\) from other \((L-1)\) users and calculate unique \(s\) based on these messages. In this process, user \(u\)'s participation is dispensable. Fig. 6 shows the procedure of system initialization.

Database construction. In PPMARS-C, database is constructed in a certain period according to system design or user configuration. Generally, the period is about 6 or 12 h in order to fetch features of user trust behaviors of mobile application usage. In PPMARS-S, however, database construction starts after a requesting user \(k\) sends a recommendation request to SP, which means that database construction is actually a step of recommendation generation and it will be conducted whenever one user who is in the user group as described in system initialization launches a recommendation request.

Take user \(k\) for an example, the procedure of database construction is described as follows. Note that user \(k\) must be a member of the \(L\) users and the “user group” herein means the \(L\) users unless there is a special indication. Firstly, A Trust Behavior Monitor in the user mobile device monitors the user's trust behaviors of mobile application usage and collects the user's data related to UB, RB, and CB automatically at background with user authorization. Then, the system obtains the values of UB, RB, and CB and calculates in the Calculator and stored in the Database of the user mobile device. Secondly, user \(k\) informs \(PK_k\) to other users in the group. Other users in the
**Fig. 5.** A procedure of interactive recommendation generation at time period $t$ in PPMARS-C.

**Fig. 6.** A procedure of system initialization in PPMARS-S.
user group receive $PK_k$ and encrypt $s$ into $E(s)$ with (user $k$’s public key) and store $E(s)$ in MD Database. Note that user $k$ also calculates and stores $E(s)$ in MD Database. Thirdly, user $k$ signs his anonymous identity $ID_k$ and sends a recommendation request $\{Sig_k(ID_k), PK_k, ID_k\}$ to SP. Finally, SP receives the recommendation request and sends data collection requests to all users in the system. Then, the $L$ users (i.e., the user group including user $k$) multiply the formalized values of UB, RB, and CB with $E(s)$ and send them together with their anonymous identities to SP. SP receives these data and stores them in SP Database. Note that the requirement of data storage for SP is not strict because the amount of user data should not be too big in this database construction. Fig. 7 shows a procedure of database construction. $T_i^f(k)$ represents the formalized value of user $j$’s Using Behavior on application $i$ at time $t$. Other similar symbols follow the same style of representation.

**Interactive recommendation generation.** This procedure is similar to the interactive recommendation generation in PPMARS-C except for that there are no recommendation request and user validity check, as is shown in Fig. 8.

For the purpose of security, when next requester wants to get recommendations, he/she will firstly issue his/her own secret and share it within the user’s group based on a secure protocol, e.g., Shamir Threshold Protocol [45]. Using anonymous identity, the scheme can protect the user’s real identity to some extent. Besides, the user’s real relationship in terms of similarity can be well protected from SP by introducing a secret multiplication on the formalized values of UB, RB, and CB. Applying homomorphic encryption on the user’s data can further prevent SP from knowing exact recommendations for a user, because the final recommendation is calculated in the user’s own device, thus protecting the privacy of personalized recommendation.

5. **Security analysis and performance evaluation**

In this section, we analyze the security of the proposed schemes based on the security model mentioned in Section 3. We implement the proposed two schemes by developing two Apps and conduct experiments to evaluate the performance of two Apps in terms of memory cost, CPU consumption, communication cost, and battery consumption. Finally, we simulate bad-mouth attack, on–off attack, and conflict behavior attack in order to test the robustness of two schemes. The results of simulations show that both of two schemes have sound robustness regarding the above three types of attacks.

5.1. **Security analysis**

As presented in Section 1, the motivation of this paper work is to design preserving-preserving mobile application recommender systems that can provide accurate and personalized mobile application recommendations while preserve user privacy in a secure way. We designed two privacy-preserving recommender schemes regarding two different application scenarios. Refer to the secu-
Proposition 1. The identities and device information of all users can be protected in both schemes.

Anonymity technology is adopted in both schemes, so that the real identity of a user cannot be obtained by any other parties. Besides, in the implementation of both schemes, we inject noise into the unique identification of each smart phone. i.e., its International Mobile Equipment Identity (IMEI). Thus, the device information can also be protected.

Proposition 2. The data confidentiality of our recommender systems is guaranteed by homomorphic encryption and public key cryptosystem.

User data are encrypted and processed by utilizing homomorphic encryption and traditional public key encryption, both of which are secure and mature technologies nowadays so that the confidentiality of user data can be well guaranteed. Therefore, the security of our schemes largely relies on the security of system design.

Proposition 3. In PPMARS-S, the secret $s$ is generated based on $(N, N)$-Secret Sharing protocol (Shamir Threshold Protocol [48]) that is a secure protocol. Shamir Threshold Protocol can guarantee the validity of users, the security of secret generation, and the detection of illegal users [48].

Proposition 4. In both two schemes, SP can only get the anonymous identity of user, the protected formalized values of UB, RB, and CB, the homomorphically encrypted user data, and the number of applications installed by a user. Except for these data, SP can get nothing valuable.

Actually, among the data mentioned above, only the number of applications installed by a user is possibly regarded as useful information. The user protects private data by using the encrypted random number $s$ in PPMARS-C or the encrypted secret $s$ in PPMARS-S before uploading these data to SP. Providing anonymous identity to SP aims to identify data provided by different users so that correlation between users can be calculated based on these data. The number of applications installed by a user can be counted based on the uploaded user trust behavior data of different applications. Because the formalized values of UB, RB, and CB are protected by the encrypted $s$ in the two schemes, the calculated user correlations $Rel_i(u_j, u_k)$ are meaningless to SP. At the same time, the encrypted user correlation $HE[HPK_{u_j}, Rel(u_j, u_k)]$ with homomorphic encryption in a user’s device is also confidential for SP. SP can get nothing useful from the computations on homomorphic encrypted user data. The data calculated by SP are responded to recommendation requestor and further calculated in the requestor’s mobile device to get final recommendations, which guarantees that only the requestor can get mobile application recommendations for himself. Thus, the two schemes can protect the privacy of recommendations and user data provided for generating the recommendations.

Proposition 5. In PPMARS-C, PC can get nothing useful except for the anonymous identity of mobile users.

In PPMARS-C, mobile users provide their anonymous identities to PC during registration, so PC knows anonymous identities of all registered users. Because the secure number $s$ is chosen by PC and encrypted by the public key of PC, so PC knows $E(s)$. However, as mentioned in our security model, there is no collusion between PC and SP, and $E(s)$ is changed with time flying, so user data can be well protected under $E(s)$. Except for what is mentioned above, PC cannot get anything useful, because all the processed data are stored and calculated in SP.

Proposition 6. Permeation of illegal users can be prevented in both of the two schemes.

In PPMARS-C, all the users need to register at PC and get $E(s)$ and signature of PC after confirmation. As for the requestors who want to get application recommendations, they must provide PC’s signature on their identity signatures, and later procedure can be continued only if the user validity check is positive. As a result, only users who have legally registered into the system can get recommendation service.
In PPMARS-S, the user group is constituted based on common interests, user willingness, or social trust relationships, which can avoid the existence of strangers who may be malicious users. Besides, the (N, N)-Secret Sharing Threshold protocol used in the procedure of system initialization can detect tricksters, i.e., illegal users. Because illegal users cannot provide correct sub-secrets, i.e., \( \{ID_x, s(ID_x)\} \). In this case, the secret \( a_0 \) cannot be uniquely determined, thus the secret \( s \) cannot be generated.

**Proposition 7.** Both of two schemes can resist the existence of users who are legal but malicious (e.g., a legal user who wants to get privacy information about others). That is to say, for a legal but malicious user \( e \) in the system, \( e \) cannot get any valuable information about other users.

In PPMARS-C, \( e \) cannot get any valuable information about other users in the process of data transmission, because a secure channel is adopted for communications in the system. User \( e \) cannot get anonymous identities of others, so that it is impossible for \( e \) to get recommendations for a user by counterfeiting the user. In the procedure of interactive recommendation generation, let’s say that \( e \) captured a recommendation request to SP sent by user \( x \) in a very low probability, i.e., \( \{ SIG(ID_x), SIGG(SIG(ID_x)) \} \). In this case, it is impossible for \( e \) to get recommendations for \( x \) by utilizing replay attack. Because even if the validity check of user \( e \) is permitted by SP, SP will return related data encrypted with \( PK_x \) (i.e., the public key of user \( x \)) to \( e \), because SP thinks that the request is sent by user \( e \). However, \( e \) cannot decrypt the data without \( SK_e \), i.e., the secret key of \( e \), so later steps cannot be processed.

In PPMARS-S, we focus on the replay attack launched by \( e \) because other situations are similar to what are described in PPMARS-C. Let’s say that \( e \) captured a recommendation request to SP sent by user \( x \) in a very low probability, i.e., \( \{ SIG(ID_x), PK_x, ID_x \} \). User \( e \) wants to get recommendations for \( x \) by utilizing replay attack. However, \( e \) cannot decrypt the encrypted user correlations responded by SP without \( SK_e \), so later steps cannot be performed, either.

### 5.2. Implementation and performance evaluation

We implemented the proposed two schemes based on the system design introduced in Section 4. Paillier homomorphic encryption, RSA public key cryptosystem and SHA-512 algorithm are utilized for implementing homomorphic encryption, public key encryption, digital signature, and so on. Concretely, for PPMARS-C, we developed an Android App for android smart phones and the App plays as the client software in Fig. 1. The compiling version of Android Operate System is 4.2.2 and the test smart phone is ZTE U9180 with a CPU of Snapdragon MSM8926, 1.2 GHz work CPU frequency, a RAM of 2 GB and a ROM of 8 GB. We implemented the designed functionalities of SP and PC in a laptop ThinkPad E431 with Intel Core i5-3230M CPU of 2.6 GHz and 4 GB RAM. Related databases are implemented by MySQL with a version of 5.5. Some screenshots of the PPMARS-C App are displayed in Fig. 9.

For PPMARS-S, we also developed an Android App for android smart phones and the App plays as the client software in Fig. 2. We further implemented the designed functionalities of SP in a laptop. The information of smart phone and laptop is the same as what is described above. Some screenshots of the PPMARS-S App are displayed in Fig. 10.

We installed two Apps in the ZTE testing smart phone, and run SP and PC which are developed by Java in the ThinkPad E431 laptop. In order to test the efficiency of two schemes, we firstly measured execution time of two schemes regarding different operations and different procedures. We further test memory cost, CPU consumption, communication cost, and battery consumption of two Apps. The result shows that the developed Apps have good performance regarding above attributes. Then, we analyze the accuracy of two schemes and compare them with our previous work [1]. Finally, we show the sound robustness of two schemes in terms of typical attacks through simulations.

#### 5.2.1. Scheme efficiency

We run the two schemes in a real usage context, respectively. The two client software Apps installed in a smart phone communicate with SP through a wireless local area network in a socket-connection way. We divided each procedure in two schemes into different steps according to different operations. The operation time of two schemes is compared in Table 2.

From Table 2, we can see that key generation and secret generation take most time, but key generation just executes once in the procedure of system initialization in two schemes, and secret generation also executes once in the procedure of system initialization in PPMARS-S. In PPMARS-S, database construction executes only once. Considering that database construction executes at the backend of the smart phone, such the short execution time of database construction is imperceptible for mobile users. Note that even though we multiply execution times with unit execution time in Table 2, for example, the execution time is calculated by “40” in the operation of correlation calculation for recommendation generation in PPMARS-C, real execution time is much shorter than this calculated time. Because multi-thread technique makes it possible that several threads can calculate data concurrently, thus speeding up the calculation time. In general, the result shows that the implemented two schemes hold satisfactory efficiency.

We further ran the two schemes in a real database context with 10 users and each user has about 15 mobile applications. We compared execution time of each procedure of two schemes, as shown in Fig. 11. It can be found that PPMARS-S spends more time than PPMARS-C in system initialization and database construction, because there are secret generation, sending and receiving sub-secret operations in PPMARS, except for key generation, data processing and uploading. But it takes less time in interactive recommendation generation in PPMARS-S compared to PPMARS-C, because recommendation request is included in the first step of database construction in PPMARS-S.

Because there are few privacy-preserving mobile application recommender systems that can be found in the literature, we selected two related works [32, 33] as benchmarks to compare with our proposed schemes. Due to different system structures and procedures designed in [32, 33], we only compared the average time of recommendation generation. As shown in Fig. 12, after ridding of differentials caused by running environment (e.g., CPU and memory size), our proposed schemes have shorter average execution time than the schemes proposed in [32, 33] considering the same number of recommended items, and PPMARS-C is the most efficient.

#### 5.2.2. Memory cost

We measured memory consumption costs of the developed two Apps by executing them for some time and compared memory consumption costs with existing two Apps, i.e., WeChat (version 6.5.7) and AppLock (Smart AppLock) (version 3.18.9). WeChat is a popular social communication App that is widely used. AppLock is an App that helps users manage installed Apps in a securely way by locking some private or sensitive Apps with passwords. The comparison of memory consumption costs is shown in Fig. 13. We can see that WeChat needs much more memory compared to other three Apps because it has various additional functions such as built-in browser and payment. Compared to AppLock that occupies about 30 MB of memory, our two Apps have less memory consumption costs. Concretely, at the beginning of running, from 0 to 30 s, we stop the Trust Behavior Monitor in two Apps, the memory costs are about 17 MB and 19 MB in PPMARS-C and PPMARS-S, respectively. From
30 s to 80 s, we start the Trust Behavior Monitor in two Apps that run as an Android Service at backend, the memory costs increase to 20 MB and 25 MB in PPMARS-C and PPMARS-S, respectively. Then, we exited two Apps at 120 s and the memory consumption of two Apps was changed to 0. Considering that the memory capacity of a smart phone is normally about 2–6 GB nowadays, the memory costs of two Apps are reasonable and acceptable.

5.2.3. CPU usage

Fig. 14 shows the comparison of CPU usage of WeChat, AppLock, and our two Apps. Fig. 14(a) shows the CPU usage of WeChat within...
Table 2
Operation time of two schemes (unit: milliseconds).

<table>
<thead>
<tr>
<th>Entities</th>
<th>Procedures</th>
<th>Operations and execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PPMA-S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PPMARS-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Operations</td>
</tr>
<tr>
<td>MD</td>
<td>System initialization</td>
<td>Key generation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Anonymous identity generation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Signature generation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RSA decryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hash</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RSA encryption</td>
</tr>
<tr>
<td>SP</td>
<td>Database construction</td>
<td>Receive and store data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recommendation generation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>System initialization</td>
<td>Key generation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Signature generation and verification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RSA encryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User verification</td>
</tr>
<tr>
<td>Recommendation requestor</td>
<td>Database construction</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: \( n \): the number of users who are participated in recommendation generation, \( m \): the number of applications used by a user in a time period, \( j \): the number of system users, \( h \): the number of applications that are recommended.

Fig. 11. Execution time comparison of two schemes regarding each procedure.

Fig. 12. Execution time.

A time period of 90 s. We send messages of texts, voice, and pictures alternately in this time period. The result shows that CPU usage becomes high whenever sending messages. Fig. 14(b) shows the CPU usage of AppLock within a time period of 90 s. We locked one App using AppLock and tried to open the locked App without and with correct password respectively. The result shows that when locking an App at 20 s, the CPU usage of AppLock becomes very high, and the CPU usage also become high when we tried to open the locked App at 48 s and 65 s. Fig. 14(c) shows the CPU usage of PPMARS-C within a time period of 360 s, including system initialization, database construction, and recommendation generation. We can see that in the computational operations in each procedure, the CPU usage become high. Fig. 14(d) shows the CPU usage of PPMARS-S within a time period of 60 s that is a procedure of system initialization. Note that secret generation and sub-secret distribution used a lot of CPU. However, these computational operations are executed only in system initialization and recommendation generation. Compared to WeChat and AppLock that occupy CPU resource
frequently, the results show that both of our two Apps have lower CPU usage compared with WeChat and AppLock even though computational operations occupy some CPU resources (up to about 70% of system CPU at peaks) in a short time period, which means that our Apps have light CPU cost.

5.2.4. Communication cost

We figured out communication costs of uploading and downloading of two Apps and compared communication costs with the scheme proposed in [32] for recommending 10 items (e.g., mobile applications) to a single user based on a database that contains 100 items’ data. The comparison is shown in Fig. 15. We observe that the communication cost in one-off recommendation is low (about 5 KB in PPMARS-C and 8 KB in PPMARS-S). And uploading communication cost is higher than downloading communication cost in both proposed schemes. Obviously, our schemes have lower communication costs than that proposed in [32].

In the interactive recommendation generation procedure of two schemes, the recommendation requestor participates in necessary operations and should be online, so the networking must be available in this procedure. As shown in our performance evaluation, however, the interactive recommendation generation procedure spends little time, which is negligible for users regarding the high Internet speed nowadays. Besides, low communication cost showed in our experiments also illustrates our schemes’ usability in a real environment.

5.2.5. Battery consumption

We ran two Apps in the testing smart phone for 12 h to observe battery consumption of two Apps. The time period f of database construction in PPMARS-C was set as 10 min. We also ran WeChat and AppLock in the same smart phone for 12 h to observe their battery consumption and compared the results with our two Apps. As shown in Fig. 16, WeChat and AppLock consumed a bit more power (consumed 8% and 10% of power, respectively) than our two Apps. PPMARS-C consumed more power (about 7%) than PPMARS-S (about 5%). This is because that database construction (i.e., user data are uploaded to SP) is conducted in a time period of 10 min in PPMARS-C while there is no database construction in PPMARS-S within the 12 h. We can see that both schemes have lower battery consumptions than WeChat and AppLock.

5.2.6. Accuracy

The proposed two schemes are based on our previous work [1]. We designed different system structures and working procedures for two schemes by introducing homomorphic encryption to secure data processing, applying anonymity technology for identity protection and public key cryptosystem and signature technology
for access control, and using Shamir Threshold Protocol [48] for malicious user detection. Therefore, the two schemes proposed in this paper do not have any accuracy loss in the whole procedures compared to the recommendation in TruBeRepec [1], except that system security and user privacy can be well guaranteed. We conducted the same experiments described in section 6.5 in [1], the result shows there is no accuracy loss while privacy preservation is provided.

5.2.7. Robustness

We test the robustness of two schemes considering the following attacks: bad-mouth attack, on–off attack, and conflict behavior attack. We installed 12–15 Apps in the smart phones of users in the simulations of the three types of attacks. The number of simulated smart phones, i.e., the number of system users, is set as 100 in the simulations. In the bad-mouth attack, attackers intentionally inflate a bad target or deflate a good target [47]. In the simulation of the bad-mouth attack, formalized values of user data (i.e., UB, RB, and CB) that belong to a bad-mouth attacker are set high for 5 target Apps in whole time periods in order to intentionally increase the recommendation rank of these Apps. In the on–off attack, attackers provide honest or dishonest data alternately [47]. In the simulation of on–off attack, formalized values of user data (i.e., UB, RB, and CB) that belong to an on–off attacker are set high for 5 target Apps in a time period in order to intentionally increase the recommendation rank of these Apps. But in the next time period, the formalized values are honestly set normal for the target 5 Apps. Conflict behavior attack means that attackers intentionally provide different data in a contradictory way, by behaving honestly for a target and dishonestly for another one [38]. In the simulation of conflict behavior attack, formalized values of user data (i.e., UB, RB, and CB) that belong to a conflict behavior attacker are set high for 5 target Apps and set normal for other Apps in whole time periods in order to intentionally increase the recommendation rank of the target 5 Apps. In our experiment, the accuracy of recommendation can be indicated by the result of recommendation, i.e., recommendation list. We affirm that an accurate recommendation result should overcome the false impact caused by Apps that are targeted by attackers as well as possible. That is to say, Apps that are targeted by attackers should appear in the low locations of recommendation

![Comparison of communication cost.](image1)

![Comparison of battery consumption.](image2)
list or should not appear in recommendation list. So, we apply the following formula to calculate the accuracy of recommendation:

\[
\text{Accuracy} = 1 - \frac{\sum_{i=1}^{n} (m + 1 - l_i)}{m \sum_{j=1}^{m} w_j},
\]

where \(n\) is the number of Apps that are targeted by attackers and recommended in recommendation list, \(l_i\) is the rank of \(i\)th App that is targeted by attackers and recommended in recommendation list (e.g., if an App is ranked at the first location in the recommendation list, the value of \(l\) is 1), \(m\) is the total number of Apps that are recommended in recommendation list, \(w_j\) is the weight of location in the recommendation list. The first location in recommendation list, i.e., \(w_1\) is \(m\), and the second location in recommendation list, i.e.,
1) Bad-mouth attack: The results of bad-mouth attack simulations of two schemes are shown in Fig. 17. The results of two schemes are similar and bad-mouthing attackers have some influence on the recommendation accuracy, but the influence does not increase with time flying. The result shows that our two schemes can resist bad-mouth attack to some extent.

2) On–off attack: The results of on–off attack simulations of two schemes are shown in Fig. 18. In PPMARS-C, the on–off attack has some influence on recommendation accuracy at the beginning. But the influence decreases with time flying. This is because the proportion of honest data is increasing with periodic database constructions in different time slots. In PPMARS-S, the influence of on–off attack on recommendation accuracy fluctuates with time flying, but the fluctuation range is acceptable. The fluctuation is caused by the procedure of database construction that executes only once when generating recommendations.

3) Conflict behavior attack: The performance of two schemes under conflict behavior attack compared with our previous work [1] is shown in Fig. 19. In PPMARS-C, the conflict behavior attack has some influence on recommendation accuracy at the beginning. But the influence becomes trivial with time flying. Compared with reference [1], PPMARS-C has better performance at the beginning, but it takes more time to overcome the impact of conflict behavior attack. In PPMARS-S, conflict behavior attack also impact recommendation accuracy, but the influence does not disappear with time flying. And similar to PPMARS-C, PPMARS-S has better performance at the beginning than our previous work [1], but the negative influence of conflict behavior attack cannot be overcome with time flying because malicious users who provide fake data cannot be detected and kicked out of the system, which is an issue we need to solve in the future.

The simulation results of two schemes under three types of attacks show that our proposed schemes have good robustness in terms of generating accurate mobile application recommendations with privacy preservation.

6. Conclusion

In this paper, we propose two schemes, i.e., PPMARS-C and PPMARS-S, for privacy-preserving mobile application recommendations based on our previous work. PPMARS-C is a system that can be deployed to realize Recommendation-as-a-service. It consists of three kinds of entities: a mobile client that monitors user trust behaviors and pre-processes these data by formalizing them, a Service Provider that stores data and calculates recommendations, and a Privacy Center that ensures the security and privacy of recommendations. Based on homomorphic encryption, anonymity technology and public key cryptosystem, the procedures for system initialization, database construction, and interactive recommendation generation were designed to allow PPMARS-C to provide accurate and personalized mobile application recommendations in a privacy-preserving manner in a cloud service environment. PPMARS-S is another system that can be applied in the context of social networking. It consists of two kinds of entities, a mobile client that is similar to the client of PPMARS-C, a Service Provider that generates recommendations with privacy preservation. Compared with PPMARS-C, different procedures of system initialization, database construction, and interactive recommendation generation were designed in PPMARS-S. Apart from the technologies used in PPMARS-C, Shamir Threshold Protocol was utilized to guarantee
the security of PPMARS-S for generating a shared secret. What's more, we implemented the proposed two schemes and developed two Apps in android smart phones. Experimental tests of two Apps in real usage contexts show that they have good efficiency, low memory costs, lightweight CPU consumption, low communication costs and battery consumption. The results of simulations on malicious attacks further show that our proposed two schemes have good robustness. But how to completely detect internal malicious users and further improve the accuracy and security of our schemes with regard to internal attacks is still an open issue, which is also a direction of our future work.

Acknowledgments

This work is sponsored by the National Key Research and Development Program of China (grant 2016YFB0800704), the NSFC (grants 61672410 and U1536202), the Project Supported by Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2016JDC-06), the 111 project (grants B08038 and B16037), and Academy of Finland (grant 308087).

References


Kun Xu received his B.Eng. degree in information security from Lanzhou University, Lanzhou, China in 2015. He is currently pursuing his master degree in information security from the School of Cyber Engineering, Xidian University, China. His research interests include recommender system, privacy preservation, and trusted computing.
Weidong Zhang is a computer scientist with interests in Information Security. He is currently an associate professor in Xidian University, China. He authored about 30 publications and co-invented more than 10 patents and patent applications.

Zheng Yan received the B.Eng. degree in electrical engineering and the M.Eng degree in computer science and engineering from the Xi'an Jiaotong University, Xi'an, China in 1994 and 1997, respectively, the second M.Eng. degree in information security from the National University of Singapore, Singapore in 2000, and the licentiate of science and the doctor of science in technology in electrical engineering from the Helsinki University of Technology (current Aalto University), Helsinki, Finland in 2005 and 2007. She is currently a full professor at the Xidian University, Xi'an, China and a visiting professor and an academy research fellow at the Aalto University, Espoo, Finland. Her research interests are in trust, security and privacy, as well as data mining. Prof. Yan is serving as an associate editor of IEEE Internet of Things Journal, IEEE Access, Information Sciences, Information Fusion, JNCA, Soft Computing, SCN, etc. twelve journals. She serves as an organization and program committee member for numerous international conferences. She is a senior member of the IEEE.