A Privacy-Preserving Framework for Trust-Oriented Point-of-Interest Recommendation

An Liu, Weiqi Wang, Zhixu Li, Guanfeng Liu, Qing Li, Xiaofang Zhou, Xiangliang Zhang

***Abstract*—Point-of-Interest (POI) recommendation has attracted many interests recently because of its significant potential for helping users to explore new places and helping LBS providers to carry out precision marketing. Compared with the user-item rating matrix in conventional recommender systems, the user-location check-in matrix in POI recommendation is usually much more sparse, which makes the notorious cold start problem more prominent in POI recommendation. Trust-oriented recommendation is an effective way to deal with this problem but it requires that the recommender has access to user check-in and trust data. In practice, however, these data are usually owned by different businesses who are not willing to share their data with the recommender mainly due to privacy and legal concerns. In this paper, we propose a privacy-preserving framework to boost data owners willingness to share their data with untrustworthy businesses. More specifically, we utilize partially homomorphic encryption to design two protocols for privacy-preserving trust-oriented POI recommendation. By offline encryption and parallel computing, these protocols can efficiently protect the private data of every party involved in the recommendation. We prove that the proposed protocols are secure against semi-honest adversaries. Experiments on both synthetic data and real data show that our protocols can achieve privacy-preserving with acceptable computation and communication cost.**

***Index Terms*—Trust, privacy, recommendation, encryption, point-of-interest.**

I. Introduction

THE rapid development of GPS-enabled mobile devices and ubiquitous Internet access has led to the boom of location-based service (LBS), which makes people much easier to share their experiences of places, also known as Point-of-Interests (POIs). For example, Foursquare announced in 2016 that it has logged more than 100 million POIs worldwide, with more than 600 million photos and 87 million tips, contributed by more than 50 million users. These check-in data reflect users' preferences for places and thus provide a solid foundation for personalized POI recommendation. This emerging kind of recommendation is significant as it not only helps users to explore new or relevant places without spending much time on searching but also enables LBS providers to carry out precision marketing for POI owners.

Although recommender systems have been comprehensively studied in the past decades [1], personalized POI recommendation [2], [3] has just attracted increasing attention recently due to its unique characteristics. Compared with the user-item rating matrix in conventional recommender systems, the user-location check-in matrix in POI recommendation is usually much more sparse. For example, the density of Gowalla dataset is 2.08×10^{-4}, while that of Netflix dataset is around 0.01 [3]. The extreme sparsity of check-in data makes it difficult to capture accurately users' preferences for places. Geographical influence is another factor that distinguishes POI recommendation from conventional item recommendation. Analysis of check-in data shows that users usually prefer visiting nearby places to those far away. In addition, a user's location history is typically locally crowded [4], which makes the cold start problem more prominent in POI recommendation. This is because even if a user has visited enough places near his/her residential area, the cold start problem will be encountered inevitably when he/she travels to some new areas.

Trust-oriented recommendation is an effective way to deal with the above cold start and data sparsity problem [5], [6]. This kind of methods assume the additional knowledge of a trust network (e.g., an online social network) among users. Due to the propagation of trust over the social network [7]–[9] it is possible to compute the trust between two users in the same social network even if they have not rated any items or visited any places in common. It is worth noting that the exploitation of trust in recommendation does not necessarily enhance the precision, but it allows to involve more users in the recommendation which in particular helps to generate good results for cold start users. Further, it makes recommendation more resilient against malicious attacks like fake profiles since these profiles are not trustworthy and could be easily identified and discarded in the course of recommendation. Therefore, trust has also been extensively explored to enhance POI recommendation [10]–[12].

To simplify discussion current approaches for trust-oriented POI recommendation commonly assume that recommender owns all the data required for computation. However, this is rarely the case in practice. To facilitate subsequent discussion, we distinguish here three types of data required in the course of POI recommendation: user check-ins, a trust network, and the current position of a target user. Generally, these data belong to different entities. More specifically, user check-ins are usually owned by an LBS provider like Foursquare, while a trust network is typically the asset of a social networking

The work described in this paper was partially supported by KAUST and the National Natural Science Foundation of China (No. 61572336, No. 61402313, No. 61702227, No. 61572335, and No.61472337).

A. Liu, W. Wang, Z. Li, G. Liu, X. Zhou are with the School of Computer Science and Technology, Soochow University, Suzhou, China.

Q. Li is with the department of Computer Science, City University of Hong Kong, Hong Kong, China.

X. Zhou is with the School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, Australia.

A. Liu and X. Zhang are with King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia.

Corresponding author: Guanfeng Liu (gfliu@suda.edu.cn)

site like Facebook. For a target user, his/her current location is his/her private data unless he/she is willing to share it with others. Clearly, a POI recommender cannot adopt existing recommendation techniques directly because it does not always have all these data in hand. This problem will be exacerbated when data owners are reluctant to share their data with the recommender due to privacy concerns.

A privacy-preserving framework can be used to boost data owners' willingness to share their data with untrustworthy entities, thus overcoming the weakness of existing approaches for trust-oriented POI recommendation. Without loss of generality, we assume that the recommender $\mathcal{R}$ is an LBS provider who wants to suggest a set of POIs to its users. In order to consume the location-based services provided by $\mathcal{R}$, the users agree to sign a contract with $\mathcal{R}$, which permits it to access their current locations, but forbids it to disclose these data to any other third party. We consider a general case where $\mathcal{R}$ does not have user check-ins and the trust network. Instead, user check-ins are hold by another LBS provider $\mathcal{L}$, while the trust network is hold by a social network site $\mathcal{S}$. The general objective of the privacy-preserving framework is therefore to enable $\mathcal{R}$ to perform recommendation while preserving private data of $\mathcal{L}$, $\mathcal{S}$, and $\mathcal{R}$.

To achieve this goal, we identify the private data and how they are involved in the computation of trust-oriented POI recommendation. We selectively encrypt some private data by using partially homomorphic encryption and perform the whole computation on the encrypted data to preserve the privacy of every party involved in the recommendation. To reduce the computation and communication cost incurred by operations over encrypted data, we design two protocols in which most expensive operations can be carried out offline or in parallel. The main contributions of our work in this paper are summarized as follows:

- We introduce a practical setting where effective POI recommendation is made by the cooperation of three parties (i.e., the recommender, LBS provider, and social networking site) while their private data are kept secret.
- We enhance current solutions to trust-oriented POI recommendation by removing the assumption that the recommender owns all the data required for computation. With the proposed protocols, the recommender can make trust-oriented POI recommendation without knowing check-in and trust data.
- We propose two protocols for privacy-preserving trust-oriented POI recommendation. Our protocols utilize partially homomorphic encryption to protect the private data of every party involved in the recommendation and their performance gets significant improvement by using offline encryption and parallel computing.
- We theoretically analyze the security of the proposed protocols in the semi-honest model. We also conduct extensive experiments to evaluate the performance of the proposed protocols by using both synthetic data and two real datasets.

The remainder of this paper is organized as follows. We give a literature view in Section II. We discuss the system model, adversary model, and problem definition in Section III. Section IV briefly introduce the cryptographic system used in our solution. Section V presents two protocols for privacy-preserving trust-oriented POI recommendation. Experimental results are given and analyzed in Section VI. Finally, Section VII concludes this paper.

II. Related Work

This study is an intersection of multiple disciplines, including POI recommendation, trusted computing, and privacy-preserving. Recently there have been several surveys on POI recommendation [2], [3], [13], which elaborate on the achievements on this topic in the past decade. We therefore do not discuss the studies on POI recommendation here and refer readers to these surveys for more details on this hot research issue. Below we will review some recent works on trusted computing and privacy-preserving recommender systems.

A. Trust Aggregation and Propagation

In the literature, existing models on trust aggregation and propagation can be classified into three categories based on the types of trust transitivity strategies they adopted: 1) multiplication strategy, 2) averaging strategy, and 3) confidence-based strategy.

In the first category, the trustworthiness of a target participant is computed as the multiplication of the trust values between any two adjacent participants along a social trust path. For example, if u_1 trusts u_2 with t_{12} and u_2 trusts u_3 with t_{23} where $t_{12}, t_{23} \in [0, 1]$, then u_1 trusts u_3 with $t_{13} = t_{12} \times t_{23}$. This strategy has been adopted in many existing models, for example, in [14] and [15].

In the second category, the trustworthiness of a target participant is computed based on averaging the trust values between any two adjacent participants along a social trust path, that is, $t_{13} = (w_{12}t_{12} + w_{23}t_{23})/2$, where w_{12} and w_{23} are the weights of t_{12} and t_{23} respectively, and $w_{12} + w_{23} = 1$. The trust transitivity models proposed in [16] and [17] belong to this category.

In the last category, the confidence between participants is considered in trust transitivity, that is, t_{13} is calculated based on t_{12}, t_{23} and the confidence of u_1 on t_{23} (denoted as f_{31}). f_{31} is computed based on the preference similarity between u_1 and u_2, and it is proportional to the latter. This strategy has been adopted in [18] and [19].

B. Privacy-preserving Recommendation

Shokri et al. present a recommender system built on distributed aggregation of user profiles, which suffers from the trade-of between privacy and accuracy [20]. In [21], [22], Nikolaenko et al. consider two basic problems in model-based recommendation algorithms: matrix factorization and ridge regression. For the first problem, they propose a solution based on Yao's protocol. For the second problem, they design a hybrid method by combing Yao's protocol and homomorphic encryption. In [23], the authors present a solution for privacy preserving recommendation via homomorphic encryption and data packing.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2765317, IEEE Access

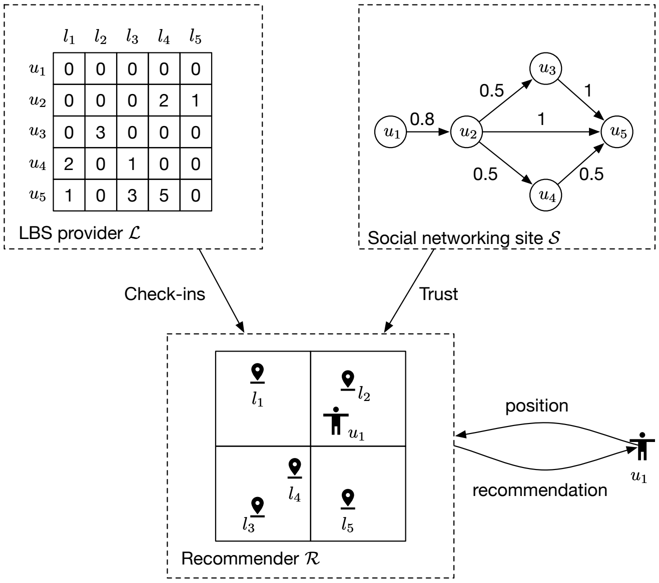
In [24], the authors propose a lightweight privacy-preserving technique called randomized perturbation. They claim that accurate recommendation could still be obtained while randomness from a specific distribution are added to the original data to prevent information leakage. A similar idea is also presented in a recent work [25], the objective of which is to realize privacy-preserving web services QoS prediction. However, the range of randomness is chosen by experience and does not have provable privacy guarantee. What's worse, it is recognized that with the application of clustering on the perturbed data, adversaries can accurately infer users' private data with accuracy up to 70% [26].

Recently there are lots of achievements in privacy preserving recommender systems under the model of differential privacy. McSherry and Mironov [27] integrate differential privacy into non-social recommender systems. However, their approach will lead to an unacceptable loss of utility when applied to social recommendation. To overcome this weakness, Jorgensen and Yu [28] incorporate a clustering procedure that groups users according to the natural community structure of the social network and significantly reduces the amount of noise. [29] proposes a distance-based differential privacy framework that computes k-NN and provides recommendations. These works are all interested in preserving the collected users' data, but ignore the privacy of user's data in the phase of data collection. In [30], the authors propose a privacy-preseving collaborative QoS prediction framework which can protect the private data of users while retaining the ability of generating accurate QoS prediction. [31] presents a hybrid approach for privacy-preserving recommender systems by combining randomized perturbation and differential privacy. Users' private data are protected by randomized perturbation and the privacy of recommendation result is guaranteed by differential privacy. Shen et al. [32] design a novel and practical privacy-built-in client under untrusted server settings, in which users' data are perturbed and anonymized on their private devices before collected by the server.

III. Problem Formulation

A. System Model

Consider a typical trust-oriented POI recommendation scenario in which the recommender $\mathcal{R}$ wants to recommend a set of POIs to its user, say v, based on his/her current location l_v, check-ins, and trust data. Before discussing the detailed recommendation procedure, we first take an overview of the data involved in recommendation from the ownership point of view, highlighting the challenges in privacy-preserving trust-oriented POI recommendation. In our model, $\mathcal{R}$ has the authority to know v's location according to the contract between them. However, check-ins and trust data are hold by the LBS provider $\mathcal{L}$ and the social networking site $\mathcal{S}$, respectively. The recommender wants to perform recommendation for v with the help of $\mathcal{L}$ and $\mathcal{S}$ while hiding v's location from them because the contract stipulates that $\mathcal{R}$ is not allowed to disclose users' locations to other parties. On the other hand, both $\mathcal{L}$ and $\mathcal{S}$ are willing to collaborate with $\mathcal{R}$ on recommendation, but either of them also wants to hide his/her private data from the other and $\mathcal{R}$. Informally, we target privacy-preserving POI recommendation in the sense that the private data of every entity involved in the recommendation should be kept secret.



	l_1	l_2	l_3	l_4	l_5
u_1	0	0	0	0	0
u_2	0	0	0	2	1
u_3	0	3	0	0	0
u_4	2	0	1	0	0
u_5	1	0	3	5	0

Fig. 1: System architecture.

Let us take a closer look at the data hold by different entities. The LBS provider $\mathcal{L}$ possesses the check-in data of n users $U_{\mathcal{L}} = \{u_1, u_2, \cdots, u_n\}$ for m locations $L_{\mathcal{L}} = \{l_1, l_2, \cdots, l_m\}$. Let $\mathbf{C}$ denote the user-location check-in matrix hold by $\mathcal{L}$ and c_{ij} the number of check-ins that user u_i has ever done at location l_j. The social networking site $\mathcal{S}$ has a trust network that is modeled by a directed graph, denoted by $G = (U_{\mathcal{S}}, E)$, which consists of a set of users, $U_{\mathcal{S}}$, and a set of edges, $E \subseteq U_{\mathcal{S}} \times U_{\mathcal{S}}$. An edge $(u, v) \in E$ represents a trust relation between two users $u, v \in U_{\mathcal{S}}$ and is associated with a weight t_{uv}, indicating the trust value that the user u assigns to the user v. Without loss of generality, we assume $t_{uv} \in [0, 1]$. If $t_{uv} = 0$, u completely distrusts v. If $t_{uv} = 1$, u completely trusts v. To simplify later discussion, we assume that $U_{\mathcal{L}} = U_{\mathcal{S}}$, that is, $\mathcal{L}$ and $\mathcal{S}$ have the same set of users. However, our solution can be easily extended to the case of $U_{\mathcal{L}} \neq U_{\mathcal{S}}$. Finally, the recommender $\mathcal{R}$ has the positions of its users and a set of locations $L_{\mathcal{R}}$ to be recommended. We also assume $L_{\mathcal{L}} = L_{\mathcal{S}}$ for ease of discussion, but again our solution can be easily extended to cope with $L_{\mathcal{L}} \neq L_{\mathcal{S}}$. From now on, we will omit the subscript of U and L unless otherwise signified.

Figure 1 shows the proposed system architecture of trust-oriented POI recommendation. A user say u_1 wants to get suggestions on where to go by sending his/her location to $\mathcal{R}$. Upon receiving u_1's location l_{u_1}, $\mathcal{R}$ starts the computation procedure of recommendation. As it does not have all the data required for making recommendation, $\mathcal{R}$ requests check-in data from $\mathcal{L}$ and trust data from $\mathcal{S}$. After that, it follows an existing recommendation algorithm (e.g., collaborative filtering that will be discussed below) to calculate the scores of all locations in L for u_1. Finally, k locations with the highest scores (a.k.a. top-k locations) are recommended to u_1.

Collaborative filtering (CF) [33] is a mature recommendation algorithm that has been widely used by commercial recommender systems (e.g., Amazon and Netflix) recently. In this paper, we assume that $\mathcal{R}$ adopts user-based CF, whose basic idea is that similar users are most likely to have the similar preferences over the same locations. As mentioned earlier, trust values are utilized during recommendation to overcome the cold start problem. Let s_{ul} denote the score of recommending location l to user u (or just the score of location l). By measuring similarity of two users based on the trust between them, s_{ul} is calculated as follows:

$$s_{ul} = \sum_{v \in U, v \neq u} t_{uv} \times c_{vl} \tag{1}$$

where t_{uv} is the trust value between u and v, and c_{vl} is the number of check-ins that v has ever done at l. It should be noted that t_{uv} will be evaluated by the methods discussed in Section II-A if u is not adjacent to v.

B. Adversary Model and Security Definition

We first list the private data in the course of trust-oriented POI recommendation for a particular user u as follows:

1) Input of $\mathcal{R}$: u's location l_u,
2) Input of $\mathcal{L}$: check-ins $\mathbf{C}$,
3) Input of $\mathcal{S}$: a trust network G.

Privacy-preserving means all the above private data should be hidden from unauthorized parties during recommendation. To accurately define the ability of unauthorized parties, we adopt a typical adversary model, i.e., the semi-honest model [34]. Specifically, all parties in this model are assumed to be semi-honest, that is, they follow the recommendation protocol exactly as specified, but may try to learn as much as possible about other parties' private input from what they see during the protocol's execution.

On one hand, the above model takes curiosity or malice into account. For example, $\mathcal{R}$ may try to determine $\mathcal{L}$'s check-in data and $\mathcal{S}$'s trust data during recommendation by using the data it receives. Similarly, $\mathcal{L}$ may try to determine u's location and $\mathcal{S}$'s trust data, and $\mathcal{S}$ may try to determine u's location and $\mathcal{L}$'s check-in data. On the other hand, honesty are also captured by this model, which is reasonable since all three parties are motivated to produce a good recommendation result. For $\mathcal{R}$, a good recommendation helps to increase its users' satisfaction. For the data providers $\mathcal{L}$ and $\mathcal{S}$, they may sell their data to more consumers by establishing a good reputation via the successful cooperation with $\mathcal{R}$. Overall, the reasonability of semi-honest model makes it widely-accepted in a variety of privacy-preserving computation domains [35]–[38].

We also assume that any two of them do not collude. As stated in [39], the assumption of non-collusion between two well-established companies (real examples for $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{S}$ are Google Place, Foursquare, and Facebook, respectively) is reasonable as the collusion will damage their reputation and consequently reduce their revenues.

We adopt the well-known real-ideal paradigm [34] to define the security of a protocol. Intuitively, a protocol is secure or privacy-preserving if every party involved in the protocol learns no more knowledge from the execution of the protocol than the knowledge that this party is entitled to know. This can be formally defined by the real-ideal paradigm as follows: for all adversaries, there exists a probabilistic polynomial-time simulator, so that the view of the adversary in the real world and the view of the simulator in the ideal world are computationally indistinguishable.

Let $\mathcal{F}(x_1, \cdots, x_n) = (\mathcal{F}_1, \cdots, \mathcal{F}_n)$ be an n-ary functionality, where x_i and $\mathcal{F}_i$ are the i-th party P_i's input and output, respectively. For $I = \{i_1, \cdots, i_t\} \subseteq [n] \overset{def}{=} \{1, \cdots, n\}$, let $\mathcal{F}_I$ denote the subsequence $\mathcal{F}_{i_1}, \cdots, \mathcal{F}_{i_t}$. Let Π be a n-party protocol for computing $\mathcal{F}$. The view of P_i during an execution of Π on $\overline{x} = \{x_1, \cdots, x_n\}$, denoted as $view_i^{\Pi}(\overline{x})$, is (x_i, r, m_i) where r represents the outcome of P_i's internal coin tosses and m_i represents the messages that it has received. For $I = \{i_1, \cdots, i_t\}$, let $view_I^{\Pi}(\overline{x}) \overset{def}{=} (I, view_{i_1}^{\Pi}(\overline{x}), \cdots, view_{i_t}^{\Pi}(\overline{x}))$. The security of $\mathscr{P}$ is formally defined as follows:

Definition 1 (Secure protocol under semi-honest model [34]). *A protocol Π privately computes $\mathcal{F}$ if there exists a probabilistic polynomial-time simulator S, such that for every $I \subseteq [n]$, it holds that:*

$$S(I, (x_{i_1}, \cdots, x_{i_t}), \mathcal{F}_I(\overline{x})) \equiv view_I^{\Pi}(\overline{x}) \tag{2}$$

where $\equiv$ denotes computational indistinguishability.

The above equation asserts that the view of every party in I can be efficiently simulated based on its input and output. That is, it cannot derive extra information during an execution of the protocol Π, indicating Π is secure or privacy-preserving.

C. Problem Statement

Below we formally define the problem of privacy-preserving trust-oriented POI recommendation (PPTR).

Definition 2 (PPTR problem). *Given a public location set $L = \{l_1, l_2, \cdots, l_m\}$, a private user-location check-in matrix $\mathbf{C}$ hold by $\mathcal{L}$, a private trust network G hold by $\mathcal{S}$, and the private location of u hold by $\mathcal{R}$, find a k-size ordered location set $L_k = \{l_{i_1}, l_{i_2}, \cdots, l_{i_k}\}$ for u, such that:*

1) *$s(u, l_i) \geq s(u, l_j)$ for $l_i \in L_k$ and $l_j \in L \setminus L_k$,*
2) *$s(u, l_{i_1}) \geq s(u, l_{i_2}) \geq \cdots \geq s(u, l_{i_k})$,*
3) *Equation (2) holds.*

In the above definition, the first requirement tells us the recommended L_k should be top-k locations while the second indicates higher demands for recommendation, that is, the recommended locations should be ranked to facilitate u's decision making about where to go. The last requirement is about security, that is, private data should be protected when computing which locations need to be recommended.

IV. Cryptographic Building Blocks

Paillier [40] is a public-key cryptosystem whose security is based on an assumption related (but not known to be equivalent) to the hardness of factoring. It consists of the following three algorithms:

TABLE I: Summary of notation

Notation	Meaning
$\mathcal{R}$	recommender
$\mathcal{L}$	LBS provider
$\mathcal{S}$	social networking site
t_{uv}	trust value between user u and user v
c_{vl}	number of check-ins that user v has ever done at location l
s_{ul}	score of recommending location l to user u
s'_{ul}	masked score of recommending location l to user u
$PK_{\mathcal{S}}$	public key of $\mathcal{S}$
$PK_{\mathcal{L}}$	public key of $\mathcal{L}$
$E_{\mathcal{S}}(x)$	encryption of message x using $PK_{\mathcal{S}}$
$E_{\mathcal{L}}(x)$	encryption of message x using $PK_{\mathcal{L}}$

- *Key generation*: Choose two distinct large random primes p, q and compute $N = pq$. Choose an element $g \in \mathbb{Z}^*_{N^2}$. The public key k_p is (N, g) and the secret key k_s is (p, q).
- *Encryption* E: Let m be a message in $\mathbb{Z}_N$. It is encrypted by selecting a random number r in $\mathbb{Z}^*_N$ and computing
$$c = E(m) = g^m r^N \mod N^2.$$
- *Decryption* D: The ciphertext c is decrypted by computing
$$m = D(c) = \frac{((c^\lambda \mod N^2) - 1)/N}{((g^\lambda \mod N^2) - 1)/N} \mod N,$$
where $\lambda = lcm(p-1, q-1)$.

One of the most important properties of Paillier is *homomorphic addition*. Specifically, multiplying an encryption of m_1 and an encryption of m_2 results in an encryption of $m_1 + m_2$, and raising an encryption of m to a constant k results in an encryption of km, that is,

$$E(m_1)E(m_2) = E(m_1 + m_2), \tag{3}$$

$$E(m)^k = E(km). \tag{4}$$

Besides, Paillier is *semantic secure*, that is, an adversary cannot learn any partial information about the plaintext from the ciphertext. As a result, it is also a probabilistic encryption scheme, which means when encrypting the same message several times, it will produce different ciphertexts.

V. Privacy-preserving Trust-Oriented POI Recommendation Protocols

In this section, we propose two protocols that can preserve the privacy of every involved party while performing effective recommendations. We first present PPTR-S in Section V-A, a secure protocol in which all data are encrypted by the public key of the social networking site $\mathcal{S}$. Its security and complexity are discussed in Section V-B and Section V-C, respectively. Finally, we present PPTR-L, another secure protocol which has the similar idea of PPTR-S, but all data in this protocol are encrypted by the public key of the LBS provider $\mathcal{L}$. In short, PPTR-S and PPTR-L achieve the same level of security, but differ in computation cost, which makes them having different application scenarios. Table I summarizes the major notations used in the protocols.

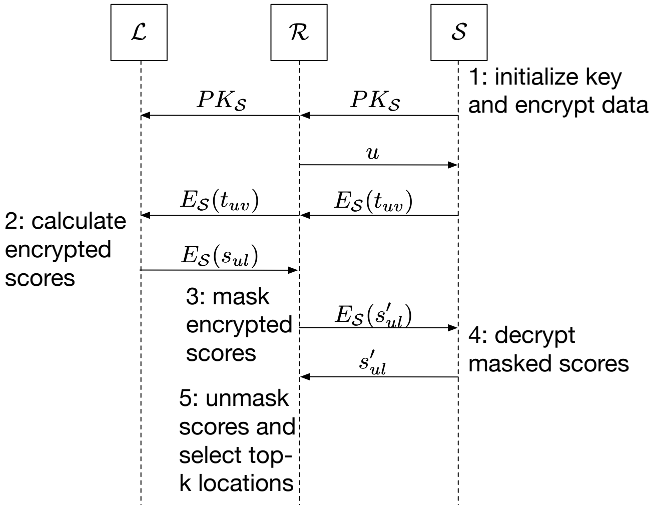

$\mathcal{L}$
$\mathcal{R}$
$\mathcal{S}$
1: initialize key and encrypt data
$PK_{\mathcal{S}}$
$PK_{\mathcal{S}}$
u
$E_{\mathcal{S}}(t_{uv})$
$E_{\mathcal{S}}(t_{uv})$
2: calculate encrypted scores
$E_{\mathcal{S}}(s_{ul})$
3: mask encrypted scores
$E_{\mathcal{S}}(s'_{ul})$
4: decrypt masked scores
s'_{ul}
5: unmask scores and select top-k locations

Fig. 2: Overview of PPTR-S protocol.

A. PPTR-S Protocol

Figure 2 shows the overview of the PPTR-S protocol. At the beginning of the protocol, the social networking site $\mathcal{S}$ generates a key pair $(PK_{\mathcal{S}}, SK_{\mathcal{S}})$ of the Paillier cryptosystem and shares the public key $PK_{\mathcal{S}}$ with the recommender $\mathcal{R}$ and the LBS provider $\mathcal{L}$. Note that there is only one key pair in the protocol, so, though all parties can perform encryption, only the holder of the private key, that is, $\mathcal{S}$, has the ability of decryption. To protect its private data G while enabling $\mathcal{R}$ to make recommendations based on trust, $\mathcal{S}$ calculates t_{uv}, the trust value between every two users $u, v \in U$, and encrypts t_{uv} using its public key $PK_{\mathcal{S}}$. Upon receiving the request from $\mathcal{R}$ which specifies the target user u, $\mathcal{S}$ sends back the encrypted trust value $E_{\mathcal{S}}(t_{uv})$ for every $v \in U \setminus u$ to $\mathcal{R}$, by which these encrypted data are forwarded to $\mathcal{L}$ for further processing. In particular, $\mathcal{L}$ calculates the encrypted score for every location $l \in L$ as follows:

$$E_{\mathcal{S}}(s_{ul}) = \prod_{v \in U \setminus u} (E_{\mathcal{S}}(t_{uv}))^{c_{vl}} \tag{5}$$

where c_{vl} is $\mathcal{L}$'s private data and $E_{\mathcal{S}}(t_{uv})$ the data provided by $\mathcal{R}$. Equation 5 is in fact equivalent to equation 1 over ciphertext, whose correctness is guaranteed by the homomorphic properties of Paillier cryptosystem, which have been shown in equations 3 and 4. The encrypted scores are then sent to $\mathcal{R}$ for deciding which locations should be recommended to u. To find top-k locations, $\mathcal{R}$ first masks the encrypted score $E_{\mathcal{S}}(s_{ul})$ for every $l \in L$ as follows:

$$E_{\mathcal{S}}(s'_{ul}) = E_{\mathcal{S}}(s_{ul}) \times E_{\mathcal{S}}(r) \tag{6}$$

where r is a random number chosen by $\mathcal{R}$. After that, it permutes these masked encrypted scores and sends them to $\mathcal{S}$ for decryption. Using its secret key, $\mathcal{S}$ decrypts $E_{\mathcal{S}}(s'_{ul})$ and obtains s'_{ul}, a masked score for every $l \in L$. Finally, $\mathcal{R}$ unmasks these masked scores by subtracting the added randomness from them and finds top-k locations for u.

We use a numerical example to illustrate an execution of the above PPTR-S protocol. The data used for computation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2765317, IEEE Access

are shown in Figure 1. Suppose that $\mathcal{R}$ wants to recommend top two POIs from $\{l_1, l_2, l_3, l_4, l_5\}$ for the user u_1. In step 1, $\mathcal{S}$ calculates the trust values between u_1 and u_i for $2 \leq i \leq 5$ and gets $t_{12} = 0.8, t_{13} = 0.4, t_{14} = 0.48$, and $t_{15} = 0.48$. Based on these encrypted values and its own check-in data, $\mathcal{L}$ computes the encrypted scores in step 2. For l_1, it performs the following computation:

$$\begin{aligned} E_{\mathcal{S}}(s_{ul_1}) &= (E_{\mathcal{S}}(t_{12}))^{c_{12}}(E_{\mathcal{S}}(t_{13}))^{c_{13}}(E_{\mathcal{S}}(t_{14}))^{c_{14}}(E_{\mathcal{S}}(t_{15}))^{c_{15}} \\ &= (E_{\mathcal{S}}(0.8))^0 E_{\mathcal{S}}(0.4))^0 E_{\mathcal{S}}(0.48))^2 E_{\mathcal{S}}(0.48))^1 \\ &= E_{\mathcal{S}}(0)E_{\mathcal{S}}(0)E_{\mathcal{S}}(0.96)E_{\mathcal{S}}(0.48) \\ &= E_{\mathcal{S}}(1.44). \end{aligned}$$

Encrypted scores for other locations are calculated similarly: $E_{\mathcal{S}}(s_{ul_2}) = E_{\mathcal{S}}(1.2), E_{\mathcal{S}}(s_{ul_3}) = E_{\mathcal{S}}(1.92), E_{\mathcal{S}}(s_{ul_4}) = E_{\mathcal{S}}(4)$, and $E_{\mathcal{S}}(s_{ul_5}) = E_{\mathcal{S}}(0.8)$. In step 3, $\mathcal{R}$ chooses five random numbers say 2, 4, 5, 1 and 3, respectively, to mask five encrypted scores. As a result, the masked values are: $E_{\mathcal{S}}(s'_{ul_1}) = E_{\mathcal{S}}(3.44), E_{\mathcal{S}}(s'_{ul_2}) = E_{\mathcal{S}}(5.2), E_{\mathcal{S}}(s'_{ul_3}) = E_{\mathcal{S}}(6.92), E_{\mathcal{S}}(s'_{ul_4}) = E_{\mathcal{S}}(5)$, and $E_{\mathcal{S}}(s'_{ul_5}) = E_{\mathcal{S}}(3.8)$. Then, $\mathcal{R}$ chooses a random permutation, say,

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 4 & 3 & 1 \end{pmatrix},$$

to permute these masked values before sending them out. Upon receiving $E_{\mathcal{S}}(3.8), E_{\mathcal{S}}(3.44), E_{\mathcal{S}}(5), E_{\mathcal{S}}(6.92)$, and $E_{\mathcal{S}}(5.2)$ from $\mathcal{R}$, $\mathcal{S}$ decrypts them and sends back these masked values. Based on the random numbers and random permutation it chosen, $\mathcal{R}$ can recover the real score of every location. In this example, it has $s_{ul_1} = 1.44, s_{ul_2} = 1.2, s_{ul_3} = 1.92, s_{ul_4} = 4$, and $s_{ul_5} = 0.8$. Clearly, l_4 and l_3 are top two locations and thus they are recommended to u_1.

Remark 1. *It is well-known that Paillier cryptosystem works on sequence of bits. On the other hand, we assumed that t_{uv} is a number between 0 and 1 in Section III-A. However, they are not contradictory because floating point numbers are typically stored in the form of a sequence of bits, for example, according to IEEE 754 standard for floating-point arithmetic [41]. Regarding implementation, it should be noted that some programming languages, such as Java, have already offered functions for doing such conversion.*

B. Security Analysis of PPTR-S Protocol

In this section, we show the security of PPTR-S protocol in the semi-honest model. Recall that any two parties in PPTR-S are not allowed to collude as stated in Section III-B. Therefore, we only need to show the view of every party in PPTR-S can be efficiently simulated based on its input and output only.

We first consider $\mathcal{L}$. During an execution of PPTR-S, $\mathcal{L}$ receives $|U| - 1$ messages in the form of $E_{\mathcal{S}}(t_{uv})$ from $\mathcal{R}$, so its view is $view_{\mathcal{L}} = \{E_{\mathcal{S}}(t_{uv}) | v \in U \setminus u\}$. To construct a probabilistic polynomial-time simulator $S_{\mathcal{L}}$ that can simulate $\mathcal{L}$'s view, we let $S_{\mathcal{L}}$ generate $|U| - 1$ random numbers uniformly distributed in $\mathbb{Z}_N$. It is easy to verify that these numbers are computationally indistinguishable from $E_{\mathcal{S}}(t_{uv})$ due to the semantic security of Paillier cryptosystem.

TABLE II: Computation cost of PPTR-S

	enc.	dec.	mul.	exp.
$\mathcal{R}$	$\lvert L\rvert$	0	$\lvert L\rvert$	0
$\mathcal{S}$	$\lvert U\rvert^2 - \lvert U\rvert$	$\lvert L\rvert$	0	0
$\mathcal{L}$	0	0	$\lvert L\rvert(\lvert U\rvert - 1)$	$\lvert L\rvert(\lvert U\rvert - 1)$

Next we show that there is a probabilistic polynomial-time simulator $S_{\mathcal{S}}$ which can also simulate $\mathcal{S}$'s view efficiently. Note that $\mathcal{S}$ is the holder of the private key and it receives $|L|$ messages in the form of $E_{\mathcal{S}}(s'_{ul})$ during an execution of PPTR-S, so its view is $view_{\mathcal{S}} = \{s'_{ul} | l \in L\}$. In this case, we let $S_{\mathcal{S}}$ generate $|L|$ random numbers uniformly distributed in $\mathbb{Z}_N$ to simulate $view_{\mathcal{S}}$ by noting that $s'_{ul} = s_{ul} + r$ where r is a random number selected by $\mathcal{R}$.

The case of $\mathcal{R}$ is a little bit complicated. First we will show that we can construct a probabilistic polynomial-time simulator $S_{\mathcal{R}}$ to simulate $\mathcal{R}$'s view efficiently based on an extra knowledge K. Then we will prove that based on the knowledge K, the probability that $\mathcal{R}$ learns the private data of $\mathcal{L}$ or $\mathcal{S}$ during an execution of PPTR-S is negligible. From the messages received by $\mathcal{R}$ during an execution of PPTR-S shown in Figure 2, it is clear that $view_{\mathcal{R}} = \{E_{\mathcal{S}}(t_{uv}) | v \in U \setminus u\} \bigcup \{s_{ul} | l \in L\}$. To simulate it, we let $S_{\mathcal{L}}$ generate $|U| - 1$ random numbers uniformly distributed in $\mathbb{Z}_N$ and a L-size set of fixed numbers, that is, $K = \{s_{ul} | l \in L\}$. It is easy to see that these views are computationally indistinguishable since Paillier cryptosystem is semantic secure. Based on the knowledge $K = \{s_{ul} | l \in L\}$, $\mathcal{R}$ can construct a linear system which consists of the following L equations:

$$\begin{cases} \prod_{i=1}^{|U|-1} x_{i1} y_{ui} = s_{u1} \\ \prod_{i=1}^{|U|-1} x_{i2} y_{ui} = s_{u2} \\ \vdots \\ \prod_{i=1}^{|U|-1} x_{i|L|} y_{ui} = s_{u|L|}. \end{cases}$$

If the above linear system has a unique solution, $\mathcal{R}$ is then able to learn the private data of $\mathcal{L}$ or $\mathcal{S}$. However, this is impossible as, when $\mathcal{R}$ does not collude with $\mathcal{L}$ or $\mathcal{S}$, the number of variables in the linear system is $|L|(|U| - 1)$ while the number of equations is only $|L|$. Therefore, we can conclude that PPTR-S protocol is secure in the semi-honest model.

C. Complexity Analysis of PPTR-S Protocol

Table II shows the computation cost of the proposed PPTR-S protocol in terms of total number of encryptions, decryptions, multiplications (over ciphertext), and exponentiations (over ciphertext) performed by different parties. We ignore computation cost on plaintext as it is negligible compared with ciphertext-related operations. As shown in Figure 2, in step 1, $\mathcal{S}$ needs to encrypt the trust values between every two users in U, which results in $|U|^2 - |U|$ encryptions. In step 2, $\mathcal{L}$ calculates $E_{\mathcal{S}}(s_{ul})$ for every location $l \in L$ according to equation 5, which needs $|L|(|U| - 1)$ exponentiations and $|L|(|U| - 1)$ multiplications. The task of masking l encrypted scores in step 3 requires $\mathcal{R}$ to encrypt l random numbers and calculate the product between l pairs of ciphertext. Moreover,

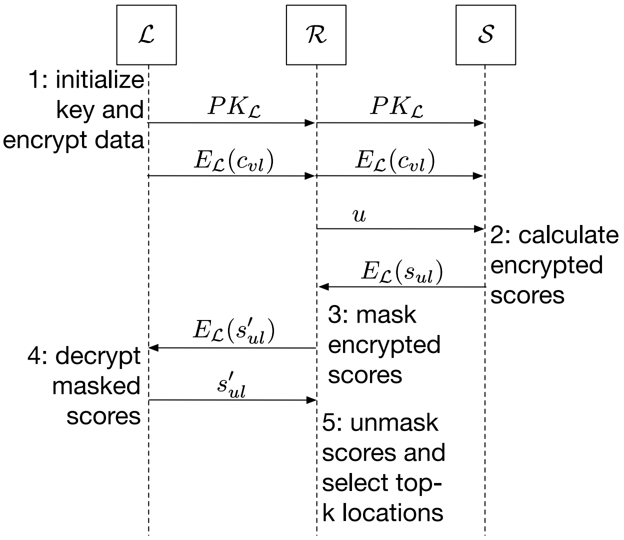

$\mathcal{L}$
$\mathcal{R}$
$\mathcal{S}$
1: initialize key and encrypt data
$PK_{\mathcal{L}}$
$PK_{\mathcal{L}}$
$E_{\mathcal{L}}(c_{vl})$
$E_{\mathcal{L}}(c_{vl})$
u
2: calculate encrypted scores
$E_{\mathcal{L}}(s_{ul})$
3: mask encrypted scores
$E_{\mathcal{L}}(s'_{ul})$
4: decrypt masked scores
s'_{ul}
5: unmask scores and select top-k locations

Fig. 3: Overview of PPTR-L protocol.

$\mathcal{S}$ needs to decrypt $|L|$ encrypted values in step 4 to enable subsequent unmasking.

From Table II, it is clear that encryption and decryption dominate the overall computation cost. Fortunately, $\mathcal{S}$ does not need to perform these encryption operations online. Instead, the trust value between every two users can be evaluated and encrypted before any request of $\mathcal{R}$ is received. This offline computation strategy also applies to $\mathcal{R}$ since the numbers for masking can be selected and encrypted before any request of $\mathcal{R}$ is sending. By offline encryption, the computation cost of PPTR-S can be reduced significantly. For the decryption, it must be performed online by $\mathcal{S}$. However, these operations are independent with each other, so they are carried out simultaneously, which further decreases the computation cost of PPTR-S.

The communication cost of PPTR-S largely comes from the ciphertexts transferred between different parties, so we only take ciphertexts into account. For $\mathcal{S}$, it sends $|U|-1$ ciphertexts to $\mathcal{R}$ and receives $|L|$ ciphertexts from $\mathcal{R}$, so its communication cost is $(|L|+|U|-1)\sigma$ where σ is key length of the Paillier cryptosystem. For $\mathcal{R}$, it: 1) receives $|U|-1$ ciphertexts from $\mathcal{S}$ and forwards them to $\mathcal{L}$; 2) receives $|L|$ ciphertexts from $\mathcal{R}$ and sends $|L|$ masked encrypted values to $\mathcal{S}$. Clearly, its communication cost is $2(|L|+|U|-1)\sigma$. For $\mathcal{L}$, it receives $|U|-1$ ciphertexts from $\mathcal{R}$ and sends $|L|$ ciphertexts to $\mathcal{R}$, so the communication cost is $(|L|+|U|-1)\sigma$. The above result is summarized in Table IV.

D. PPTR-L Protocol

The encrypted score of recommending location l to user u, s_{ul}, can be computed as shown in equation 5, which requires the public key used for encryption to be generated by $\mathcal{S}$. This is exactly what we have done in PPTR-S. In fact, s_{ul} can also be encrypted by $\mathcal{L}$'s public key as follows:

$$E_{\mathcal{L}}(s_{ul}) = \prod_{v \in U \setminus u} (E_{\mathcal{L}}(c_{vl}))^{t_{uv}} \tag{7}$$

TABLE III: Computation cost of PPTR-L

	enc.	dec.	mul.	exp.
$\mathcal{R}$	$\|L\|$	0	$\|L\|$	0
$\mathcal{S}$	0	0	$\|L\|(\|U\|-1)$	$\|L\|(\|U\|-1)$
$\mathcal{L}$	$\|L\|\|U\|$	$\|L\|$	0	0

TABLE IV: Communication cost of PPTR-S and PPTR-L

	$\mathcal{R}$	$\mathcal{S}$	$\mathcal{L}$
PPTR-S	$2(\|L\|+\|U\|-1)\sigma$	$(\|L\|+\|U\|-1)\sigma$	$(\|L\|+\|U\|-1)\sigma$
PPTR-L	$2\|L\|(\|U\|+1)\sigma$	$\|L\|(\|U\|+1)\sigma$	$\|L\|(\|U\|+1)\sigma$

which motivates us to design another protocol, PPTR-L, where $\mathcal{L}$ is responsible for generating the public key. Figure 3 gives an overview of this protocol. In step 1, $\mathcal{L}$ generates a key pair $(PK_{\mathcal{L}}, SK_{\mathcal{L}})$ of the Paillier cryptosystem, encrypts its private data, the user-location check-in matrix $\mathbf{C}$, and sends the public key and the encrypted $\mathbf{C}$ to other two parties. When $\mathcal{R}$ decides to make a recommendation to u, it asks $\mathcal{S}$ for the score of recommending l to u which is calculated by $\mathcal{S}$ according to equation 7. The returned encrypted scores are then masked by $\mathcal{R}$ in step 3, decrypted by $\mathcal{L}$ in step 4, and unmasked by $\mathcal{R}$ in step 5 using the same technique introduced in PPTR-S, so we omit the details here.

1) Security Analysis of PPTR-L Protocol: The security of PPTR-L protocol in the semi-honest model can be proved similarly as PPTR-S by exchanging the role of $\mathcal{L}$ and $\mathcal{S}$, so we do not give the details here.

2) Complexity Analysis of PPTR-L Protocol: Table III shows the computation cost of the proposed PPTR-L protocol. It costs $\mathcal{L}$ $|L||U|$ encryptions to encrypt all elements in $\mathbf{C}$ and $|L|$ decryptions to decrypt all masked values. To mask encrypted scores, $\mathcal{R}$ encrypts $|L|$ numbers and does $|L|$ multiplications over ciphertext. According to equation 7, $\mathcal{S}$ needs to perform $|L|(|U|-1)$ exponentiations and $|L|(|U|-1)$ multiplications over ciphertext. Clearly, encryption and decryption still dominate the overall cost, but the offline computation strategy for encryption and the parallel computing technique for decryption discussed earlier can be also applied here. Table IV shows the communication cost of PPTR-L. $\mathcal{L}$ sends $|L||U|$ encrypted check-ins to $\mathcal{R}$ and receives $|L|$ encrypted scores from $\mathcal{R}$, so its communication cost is $|L|(|U|+1)\sigma$. $\mathcal{S}$ incurs the same amount of cost as $\mathcal{L}$ since it receives all encrypted check-ins and sends all encrypted scores. The communication cost of $\mathcal{L}$ is clearly the sum of the cost suffered by $\mathcal{S}$ and $\mathcal{R}$ due to the symmetry of communication, which is therefore $2|L|(|U|+1)\sigma$.

VI. Performance Evaluation

A. Experimental Setting

To the best of our knowledge, there is no existing work dealing with the problem of privacy-preserving trust-oriented POI recommendation, where the data used for recommendation are not possessed solely by any single party, but are distributed among three parties. Therefore, we only evaluate and report the performance of the proposed protocols in this paper, that is, PPTR-S and PPTR-L. The performance metrics include:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2765317, IEEE Access

TABLE V: Summary of areas based on real datasets

Gowalla			Yelp		
Width (km)	#User	#POI	Width (km)	#User	#POI
5	45	82	5	612	150
10	153	241	10	1610	367
15	340	633	15	2812	576
20	897	1639	20	6120	1201

1) *CPU time*. The CPU time of a protocol is defined to be the sum of CPU time of every party involved in the protocol. Note that the simple sum is reasonable here because the computation tasks performed by different parties in PPTR-S and PPTR-L are actually in a sequential order. Moreover we only consider online CPU time. In other words, the CPU time of all computation tasks that can be done offline are not counted here.
2) *Communication cost*. The communication cost of a protocol is defined to be the amount of data that need to be transferred over some kind of network during the execution of the protocol. We evaluate this cost as it will lead to some amount of network time of data transmission. We do not evaluate the communication time directly as it is network-dependent. We consider only online communication cost like what we do for CPU time.

We use both synthetic and real data to test our proposed protocols. For synthetic data, we generate a random user-location check-in matrix with 1,000 users and 1,000 locations and a random trust network with 1,000 users. Besides, we use two real datasets, Gowalla[1] and Yelp[2]. Specifically, we select four square areas in Gowalla and four square areas in Yelp for performance evaluation. All areas in Gowalla have the same left-bottom corner, whose latitude and longitude is (33.720183,-118.399999). Similarly, we use (33.205308,-112.400283) as the left-bottom corner of all areas in Yelp. Table V shows the statistics of these areas, including width, number of users, and number of POIs.

For synthetic data, we set the number of users from 100 to 1,000 with a step of 100, and the default value is 1,000. The same setting applies to the number of POIs. For Paillier cryptosystem, we use a public available Java implementation[3]. For security parameter of Paillier, we refer to NIST recommendations (2016)[4] and set the key length to be 1024 as this value is appropriate for current applications. All experiments are performed on a PC with 3.2GHz CPU, 8GB RAM, JDK 7, and Windows10. To test the performance of parallel computing, we use a server with 40 2.4GHz CPUs which supports 40 threads.

B. Experiments on Synthetic Data

1) Effects of the number of users: Figure 4 illustrates the CPU time of the proposed two protocols by varying the number of users from 100 to 1,000 where the number of POIs is set to the default value 1,000. When the number of users increases, it is expected that the CPU time of both protocols increase almost linearly, since the number of online encryption and decrytion operations are linear with the number of users. Meanwhile, we can see that PPTR-S and PPTR-L nearly have the same amount of CPU time. This is because they need the same number of operations over ciphertext. Figure 5 shows the communication cost of the proposed two protocols by varying the number of users from 100 to 1,000 where the number of POIs is set to the default value 1,000. As the number of users becomes large, the communication cost of PPTR-S becomes large while that of PPTR-L remains the same. The reason is that the data that needs to be transferred online during the execution of PPTR-L is independent of the number of users.

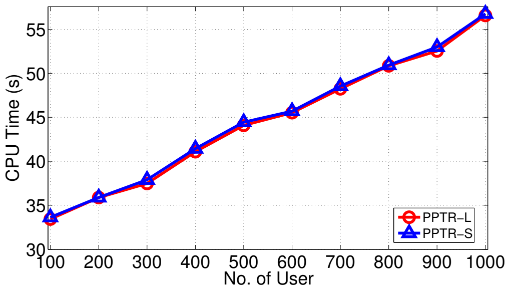


Fig. 4: Effect of number of users on CPU time.

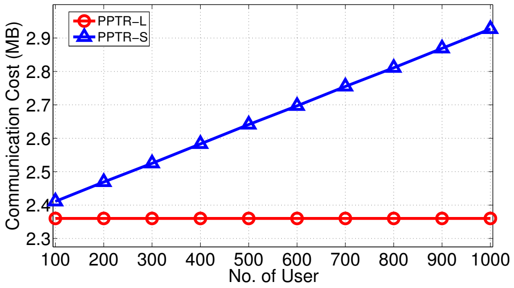


Fig. 5: Effect of number of users on communication cost.

2) Effects of the number of POIs: Figure 6 illustrates the CPU time of the proposed two protocols by varying the number of POIs from 100 to 1,000 where the number of users is set to the default value 1,000. Similar to previous results about the effect of users, the CPU time of both protocols increase almost linearly as the number of users increases. Further, PPTR-S has almost the same performance as PPTR-L in terms of CPU time, which is clear according to our theoretical analysis. Figure 7 shows the communication cost of the proposed two protocols by varying the number of POIs from 100 to 1,000 where the number of users has the default value 1,000. In this case, the communication cost of PPTR-S and PPTR-L both become large with the increase of the number of users, but PPTR-L is better than PPTR-S since it always has lower communication cost.

C. Experiments on Real Data

1) Performance on Gowalla: Figure 8 depicts the CPU time of the proposed two protocols on Gowalla dataset by varying

[1] https://snap.stanford.edu/data/loc-gowalla.html

[2] https://www.yelp.com/dataset_challenge

[3] http://www.csee.umbc.edu/k̃unliu1/research/Paillier.html

[4] https://www.keylength.com/

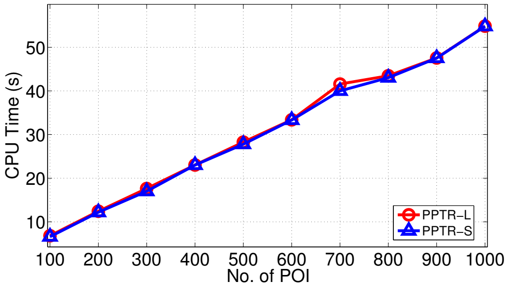


Fig. 6: Effect of number of POIs on CPU time.

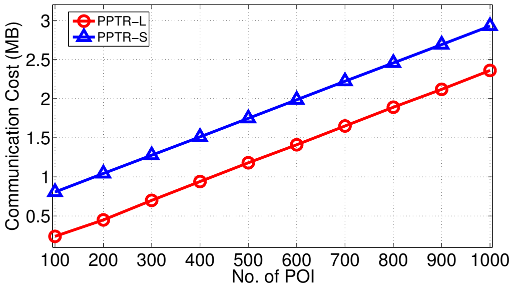


Fig. 7: Effect of number of POIs on communication cost.

the width of areas from 5 km to 20 km. When the width of areas gets large, the number of users and POIs located in the area also becomes large (see Table V), which leads to the increase of the CPU time of both protocols. Again, we observe that PPTR-S and PPTR-L nearly have the same amount of CPU time. For the 10km×10km area, both protocols need less than 10 seconds for making recommendation. Figure 9 shows the communication cost of the proposed two protocols on Gowalla dataset by varying the width of areas from 5 km to 20 km. Similar to the results about CPU time, both protocols incur more communication cost as the areas becomes larger. Moreover, PPTR-L is slightly better than PPTR-S, which shows again the superiority of PPTR-L that has been observed over synthetic data.

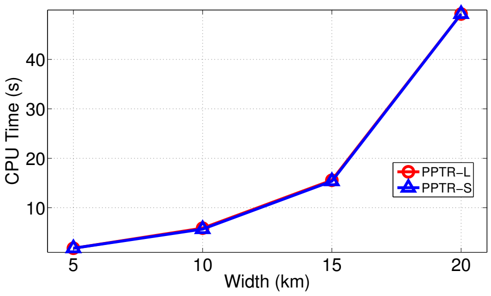


Fig. 8: CPU time on Gowalla dataset

2) Performance on Yelp: Figure 10 illustrates the CPU time of the proposed two protocols on Yelp dataset by varying the width of areas from 5 km to 20 km. Similar to previous results on Gowalla dataset, the CPU time of both protocols increase as the width of areas increases. This dataset also witnesses again PPTR-S and PPTR-L need the same amount of CPU time as these two curves almost overlap with each other. Figure 11 shows the communication cost of the proposed two protocols on Yelp dataset by varying the width of areas from 5 km to 20 km. Here, we observe that the communication cost of PPTR-S and PPTR-L both become large with the increase of the width of areas, but PPTR-L is much better than PPTR-S. This is because the number of users is much larger than the number of POIs for each area we selected on Yelp dataset and the communication cost of PPTR-L does not depend on the number of users.

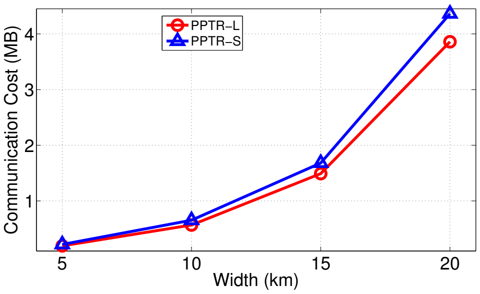


Fig. 9: Communication cost on Gowalla dataset

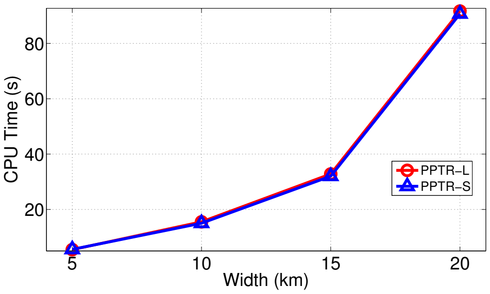


Fig. 10: CPU time on Yelp dataset

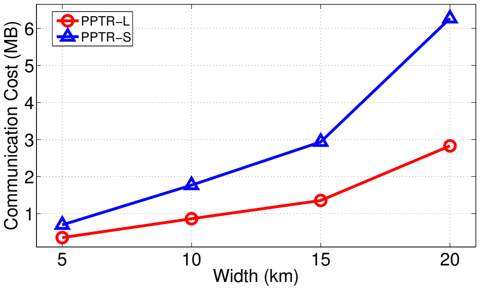


Fig. 11: Communication cost on Yelp dataset

3) Effects of parallel computing: As mentioned earlier, the expensive operations including encryption, decryption, multiplication and exponentiation over ciphertext can be parallelized since they are performed on independent data. Therefore we can greatly improve the efficiency of the proposed two protocols by using parallel computing. Figure 12 and Figure 13 depict the effects of applying parallel computing to the proposed protocols. For Gowalla dataset, in the normal case without parallel computing (i.e., one thread), both protocols need about 46 seconds for the 20km×20km area. When 16 threads are used, however, both protocols only need 4.1 seconds, which demonstrates a 11x speed up. For the 20km×20km area in Yelp dataset, both protocols need about 87 seconds when only one thread is used, but 6.9 seconds when 16 threads are used, demonstrating a 12.6x speed up. Therefore we can conclude that the proposed protocols have a good scalability by using parallel computing.

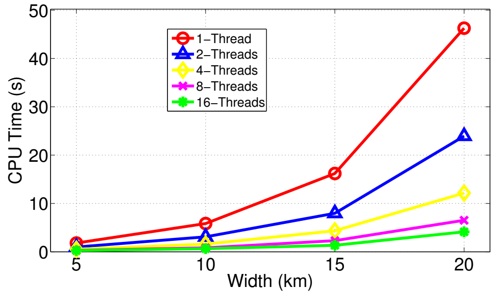


Fig. 12: Effects of parallel computing on Gowalla dataset

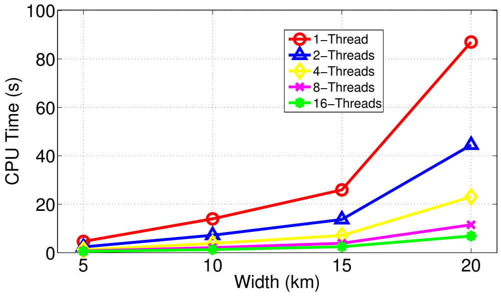


Fig. 13: Effects of parallel computing on Yelp dataset

VII. Conclusion

It is challenging to perform trust-oriented POI recommendation when the check-in and trust data are not owned by the recommender. To address this problem, we have presented a privacy-preserving framework in which the POI recommender cooperates with an LBS provider (e.g., Foursquare) who has check-in data and a social networking site (e.g., Facebook) who has trust data. This setting is more practical and enhances current solutions to trust-oriented POI recommendation by removing the assumption that the recommender owns all the data required for computation. We have utilized partially homomorphic encryption to design two protocols, PPTR-S and PPTR-L, for privacy-preserving trust-oriented POI recommendation. In PPTR-S, all data are encrypted by the public key of the social networking site who is the only one that has access to the private key. In PPTR-L, the LBS provider is in charge of generating the key pair and keeping the private key secret. These two protocols have similar computation cost but different communication cost and can be selected for use in practice according to the specific problem setting, that is, the number of users and POIs in the system. Using offline encryption and parallel computing, our protocols only needs less than 7 seconds to make recommendation for a 20km×20km area based on real data. In summary, our protocols are secure, efficient, and can scale to large POI recommendation problems in practice.

References

[1] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” *IEEE transactions on knowledge and data engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[2] J. Bao, Y. Zheng, D. Wilkie, and M. Mokbel, “Recommendations in location-based social networks: a survey,” *GeoInformatica*, vol. 19, no. 3, pp. 525–565, 2015.

[3] Y. Yu and X. Chen, “A survey of point-of-interest recommendation in location-based social networks,” in *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, vol. 130, 2015.

[4] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil, “An empirical study of geographic user activity patterns in foursquare.” *ICwSM*, vol. 11, pp. 70–573, 2011.

[5] P. Massa and P. Avesani, “Trust-aware recommender systems,” in *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 2007, pp. 17–24.

[6] M. Jamali and M. Ester, “Trustwalker: a random walk model for combining trust-based and item-based recommendation,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 397–406.

[7] A. Liu, Q. Li, L. Huang, and S. Wen, “Shapley value based impression propagation for reputation management in web service composition,” in *Web Services (ICWS), 2012 IEEE 19th International Conference on*. IEEE, 2012, pp. 58–65.

[8] A. Liu, Q. Li, X. Zhou, L. Li, G. Liu, and Y. Gao, “Rating propagation in web services reputation systems: A fast shapley value approach,” in *International Conference on Database Systems for Advanced Applications*. Springer, 2014, pp. 466–480.

[9] G. Liu, A. Liu, Y. Wang, and L. Li, “An efficient multiple trust paths finding algorithm for trustworthy service provider selection in real-time online social network environments,” in *Web Services (ICWS), 2014 IEEE International Conference on*. IEEE, 2014, pp. 121–128.

[10] C. Cheng, H. Yang, I. King, and M. R. Lyu, “Fused matrix factorization with geographical and social influence in location-based social networks.” in *Aaai*, vol. 12, 2012, pp. 17–23.

[11] J.-D. Zhang and C.-Y. Chow, “Geosoca: Exploiting geographical, social and categorical correlations for point-of-interest recommendations,” in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 443–452.

[12] H. Li, Y. Ge, R. Hong, and H. Zhu, “Point-of-interest recommendations: Learning potential check-ins from friends.” in *KDD*, 2016, pp. 975–984.

[13] S. Zhao, I. King, and M. R. Lyu, “A survey of point-of-interest recommendation in location-based social networks,” *CoRR*, vol. abs/1607.00647, 2016. [Online]. Available: http://arxiv.org/abs/1607.00647

[14] F. Walter, S. Battiston, and F. Schweitzer, “A model of a trust-based recommendation system on a social network,” *AAMAS Journal*, vol. 16, no. 1, pp. 57–74, February 2008.

[15] L. Li, Y. Wang, and E. Lim, “Trust-oriented composite services selection and discovery,” in *ICSOC’09*, 2009, pp. 50–67.

[16] E. Gray, J. Seigneur, Y. Chen, and C. Jensen, “Trust propagation in small world,” in *iTrust’03*, 2003, pp. 239–254.

[17] J. Golbeck and J. Hendler, “Inferring trust relationships in web-based social networks,” *ACM Transactions on Internet Technology*, vol. 6, no. 4, pp. 497–529, 2006.

[18] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, “Propagation of trust and distrust,” in *WWW’04*, 2004, pp. 403–412.

[19] U. Kuter and J. Golbeck, “Sunny: A new algorithm for trust inference in social networks using probabilistic confidence model,” in *AAAI’07*, 2007, pp. 1377–1382.

[20] R. Shokri, P. Pedarsani, G. Theodorakopoulos, and J.-P. Hubaux, “Preserving privacy in collaborative filtering through distributed aggregation of offline profiles,” in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 157–164.

[21] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, “Privacy-preserving matrix factorization,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 801–812.

[22] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, “Privacy-preserving ridge regression on hundreds of millions of records,” in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 334–348.

[23] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, “Generating private recommendations efficiently using homomorphic encryption and data packing,” *IEEE transactions on information forensics and security*, vol. 7, no. 3, pp. 1053–1066, 2012.

[24] H. Polat and W. Du, “Privacy-preserving collaborative filtering using randomized perturbation techniques,” in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 2003, pp. 625–628.

[25] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, “A privacy-preserving qos prediction framework for web service recommendation,” in *Web Services (ICWS), 2015 IEEE International Conference on*. IEEE, 2015, pp. 241–248.

[26] S. Zhang, J. Ford, and F. Makedon, “Deriving private information from randomly perturbed ratings,” in *Proceedings of the 2006 SIAM International Conference on Data Mining*. SIAM, 2006, pp. 59–69.

[27] F. McSherry and 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*. ACM, 2009, pp. 627–636.

[28] Z. Jorgensen and T. Yu, “A privacy-preserving framework for personalized, social recommendations.” in *EDBT*, 2014, pp. 571–582.

[29] R. Guerraoui, A.-M. Kermarrec, R. Patra, and M. Taziki, “D 2 p: distance-based differential privacy in recommenders,” *Proceedings of the VLDB Endowment*, vol. 8, no. 8, pp. 862–873, 2015.

[30] S. Liu, A. Liu, Z. Li, G. Liu, J. Xu, L. Zhao, and K. Zheng, “Privacy-preserving collaborative web services qos prediction via differential privacy,” in *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data*. Springer, 2017, pp. 200–214.

[31] X. Liu, A. Liu, X. Zhang, Z. Li, G. Liu, L. Zhao, and X. Zhou, “When differential privacy meets randomized perturbation: A hybrid approach for privacy-preserving recommender system,” in *International Conference on Database Systems for Advanced Applications*. Springer, 2017, pp. 576–591.

[32] Y. Shen and H. Jin, “Epicrec: Towards practical differentially private framework for personalized recommendation,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 180–191.

[33] Y. Shi, M. Larson, and A. Hanjalic, “Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges,” *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 3, 2014.

[34] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[35] A. Liu, K. Zheng, L. Li, G. Liu, L. Zhao, and X. Zhou, “Efficient secure similarity computation on encrypted trajectory data,” in *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE, 2015, pp. 66–77.

[36] S. Liu, A. Liu, L. Zhao, G. Liu, Z. Li, P. Zhao, K. Zheng, and L. Qin, “Efficient query processing with mutual privacy protection for location-based services,” in *International Conference on Database Systems for Advanced Applications*. Springer, 2016, pp. 299–313.

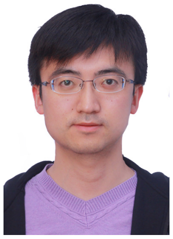
[37] L. Li, A. Liu, Q. Li, G. Liu, and Z. Li, “Privacy-preserving collaborative web services qos prediction via yaos garbled circuits and homomorphic encryption,” vol. 15, pp. 203–225, 07 2016.

[38] A. Liu, W. Wang, S. Shang, Q. Li, and X. Zhang, “Efficient task assignment in spatial crowdsourcing with worker and task privacy protection,” *GeoInformatica*, Aug 2017. [Online]. Available: https://doi.org/10.1007/s10707-017-0305-2

[39] B. K. Samanthula, F.-Y. Rao, E. Bertino, and X. Yi, “Privacy-preserving protocols for shortest path discovery over outsourced encrypted graph data,” in *Information Reuse and Integration (IRI), 2015 IEEE International Conference on*. IEEE, 2015, pp. 427–434.

[40] P. Paillier *et al.*, “Public-key cryptosystems based on composite degree residuosity classes,” in *Eurocrypt*, vol. 99. Springer, 1999, pp. 223–238.

[41] D. Zuras, M. Cowlishaw, A. Aiken, M. Applegate, D. Bailey, S. Bass, D. Bhandarkar, M. Bhat, D. Bindel, S. Boldo *et al.*, “Ieee standard for floating-point arithmetic,” *IEEE Std 754-2008*, pp. 1–70, 2008.



An Liu is an Associate Professor in the Department of Computer Science & Technology at Soochow University. He received his Ph. D. degree in computer science from both City University of Hong Kong (CityU) and University of Science and Technology of China (USTC) in 2009. His research interests include spatial databases, crowdsourcing, recommender systems, data security and privacy, and cloud/service computing. He has published more than 80 papers in referred journals and conferences, including IEEE TKDE, IEEE TSC, GeoInformatica, KAIS, ICDE, WWW, CIKM etc. He served as the Workshop Co-Chairs of WISE 2017 and DASFAA 2015. He is on the reviewer board of several top journals such as IEEE TKDE, IEEE TSC, IEEE TII, IEEE TCC, ACM TOIT, JSS, DKE, FGCS, WWWJ, and JCST. He has received the Best Paper Award from prestigious IEEE ICEBE 2012 and APWEB-WAIM 2017, and the Best Paper Award Runner Up from DASFAA 2017 and WISE2015.

Weiqi Wang is currently a master student in the Department of Computer Science and Technology at Soochow University. His main research interests include data privacy, crowdsourcing, and recommender systems. He has worked as an assistant data engineer in the company of ele.me and participated in the development of hot-food recommender system and OD-graph system.

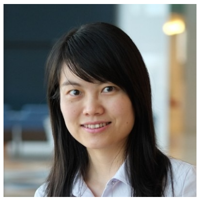
Zhixu Li is an Associate Professor in the Department of Computer Science and Technology at Soochow University, Suzhou. He used to work as a research fellow at King Abdullah University of Science and Technology. He received his Ph.D. degree in computer science from the University of Queensland in 2013, and his B.S. and M.S. degree in computer science from Renmin University of China, Beijing in 2006 and 2009 respectively. His research interests include data cleaning, big data applications, information extraction and retrieval, machine learning, deep learning, knowledge graph and crowdsourcing.

Guanfeng Liu is an Associate Professor in the Advanced Data Analytics Lab (ADA), Soochow University, China. He received his Ph.D degree from Macquarie University (MQ), Australia. Prior to joining Soochow University in 2013, he was a Postdoctoral Fellow/Research Fellow in the Department of Computing, Macquarie University.Dr Guanfeng Liu has published more than 40 papers in prestigious international journals and conferences, including IEEE TKDE, IEEE TSC, FGCS, AAAI, ICDE, ICWS, ICSOC. His research interests cover trust management, social computing, and computer security. He has served as the PC member of over 10 conferences and the reviewer of over 15 international journals. He has received the Best Paper Award from prestigious IEEE SCC2010 and APWEB-WAIM 2017, and the Best Paper Award Runner Up from WISE2015.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2765317, IEEE Access

Qing Li (SM07) received the B.Eng. degree from Hunan University, Changsha, China, and the M.Sc. and Ph.D. degrees from the University of Southern California, Los Angeles, all in computer science. He is currently a Professor with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong. His current research interests include dynamic object modeling, multimedia and mobile information retrieval and management, distributed databases and data warehousing/mining, and workflow management and web services.

Xiaofang Zhou is a Professor of computer science with The University of Queensland. He is the head of the Data and Knowledge Engineering Research Division. He is a specially appointed adjunct professor under the Chinese National Qianren Scheme hosted by the Renmin University of China (2010-2013), and by Soochow University since July 2013, where he leads the Research Center on Advanced Data Analytics. He has been working in the area of spatial and multimedia databases, data quality, high performance query processing, Web information systems, and bioinformatics, co-authored more than 250 research papers with many published in top journals and conferences. He is a senior member of the IEEE.



Xiangliang Zhang is an Associate Professor and directs the Machine Intelligence and kNowledge Engineering (MINE) Laboratory in King Abdullah University of Science and Technology (KAUST). She earned her Ph.D. degree in computer science from INRIA-University Paris-Sud 11, France, in July 2010. Her main research interests and experiences are in diverse areas of machine learning and data mining. She has published over 70 papers in referred journals and conferences, including TKDE, SIGKDD, VLDB J, AAAI, IJCAI, ICDM, ECML/PKDD, CIKM, InfoCom etc.