Fall 2013

Privacy-preserving friend recommendations in online social networks

Bharath Kumar Samanthula

Department: Computer Science

PRIVACY-PRESERVING FRIEND RECOMMENDATIONS IN ONLINE SOCIAL NETWORKS

by

BHARATH KUMAR SAMANTHULA

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2013

Dr. Wei Jiang, Advisor
Dr. Sanjay Madria
Dr. Bruce M. McMillin
Dr. Dan Lin
Dr. Akim Adekpedjou

ABSTRACT

Online social networks, such as Facebook and Google+, have been emerging as a new communication service for users to stay in touch and share information with family members and friends over the Internet. Since the users are generating huge amounts of data on social network sites, an interesting question is how to mine this enormous amount of data to retrieve useful information. Along this direction, social network analysis has emerged as an important tool for many business intelligence applications such as identifying potential customers and promoting items based on their interests. In particular, since users are often interested to make new friends, a friend recommendation application provides the medium for users to expand his/her social connections and share information of interest with more friends. Besides this, it also helps to enhance the development of the entire network structure.

The existing friend recommendation methods utilize social network structure and/or user profile information. However, these methods can no longer be applicable if the privacy of users is taken into consideration. This work introduces a set of privacy-preserving friend recommendation protocols based on different existing similarity metrics in the literature. Briefly, depending on the underlying similarity metric used, the proposed protocols guarantee the privacy of a user's personal information such as friend lists. These protocols are the first to make the friend recommendation process possible in privacy-enhanced social networking environments.

Also, this work considers the case of outsourced social networks, where users' profile data are encrypted and outsourced to third-party cloud providers who provide social networking services to the users. Under such an environment, this work proposes novel protocols for the cloud to do friend recommendations in a privacy-preserving manner.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Dr. Wei Jiang and it is a great honor for me to be his first Ph.D. student. I still remember my initial days at Missouri S&T having difficulty choosing between Master's and Ph.D. I am very grateful to Dr. Jiang for motivating me and clarifying my confusions which eventually inspired me to pursue a Ph.D. degree. Over the past five years, Dr. Jiang inspired me to learn about a wide-range of topics and supported me to work on new ideas across multiple domains. His encouragement, both academically and personally, has helped me a lot to nurture myself into an independent and positive-thinking person.

I am particularly thankful to Dr. Luo Si and Lei Cen at the Purdue University for their initial discussions on the friend recommendation problem. For the dissertation, I would like to thank my other Ph.D. committee members Dr. Sanjay Madria, Dr. Bruce M. McMillin, Dr. Dan Lin, and Dr. Akim Adekpedjou for their time and valuable suggestions. I appreciate my lab mates Yousef Elmehdwi, Michael Howard, Hu Chun, Feng Li, Prakash Kumar, and Wenquan Wang for their help at different stages of my Ph.D. journey. Also, I want to thank the staff members of different departments and the graduate editing services for their great support.

My time at Missouri S&T was filled with many memorable moments. I am grateful for time spent with my roommates Suhas Nerella and Ravi Arvapally, for my junior and senior level friends with whom I shared inspiring and memorable trips covering from New York to California. Lastly, but most importantly, I would like to thank my family members for all their support and love over the years. I am fortunate to have such a beautiful parents who supported me in all my pursuits. Also, for my kind brother Sandeep Samanthula who is currently pursuing his Master's degree at RWTH, Aachen, Germany. Thank you all.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

An online social network (OSN) facilitates users to stay in touch with other users (such as distant family members or friends), easily share information, look for old acquaintances and establish friendship with new users based on shared interests. The wide availability of the Internet has resulted in the fast growth of OSNs [1, 2, 3, 4, 5], such as Google+, Facebook, MySpace, Twitter and LinkedIn, which resulted in a vast amount of social data containing personal and sensitive information about individual users. Social network analyses [4, 6] involve mining the social data to understand user activities and to identify the relationships among various users. Especially, in applications such as business intelligence, social network analyses have boosted the research in developing various recommendation algorithms [7, 8]. For example, an algorithm may recommend a new application to a Facebook user based on either the applications he/she used in the past or the usage pattern of various applications used by his/her friends. In general, a recommendation can be a friend, a product, an ad or even a content potentially relevant to the user. This work focuses on recommending new friends to a given user in an OSN.

1.1. MOTIVATION

In particular, the friend recommendation application [9] has gained special importance from both the social network administrators and the users. From the network administrator perspective, recommending potential candidates as new friends to users will enable the development of the entire network/community since it results in a more connected network. On the other hand, from the users' side, friend recommendations help them grow their social contacts and explore for new friends based on their own interests. In general, the main goal of a friend recommendation algorithm

is to identify the potential candidates for a given target user (who wish to make new friends) in an effective manner. Along this direction, much work has been done in developing various friend recommendation algorithms [9, 10, 11, 12, 13, 14] based on network topology, user contents or both. In the literature, discovering new friends to a given user A is equivalent to solving the link prediction [15] problem for A in the corresponding social network. Given a snapshot of the social network, the link prediction problem aims at inferring the new interactions that are likely to happen among its nodes. More specifically, the nodes of the social network are the users and an edge between two users indicates a friendship between them.

Given the snapshot of an OSN, the social closeness between two users is termed as proximity. The proximity measures can be divided into two groups. The first group includes measures based on node neighborhoods, such as common neighbors, Jaccard Coefficient, Adamic/Adar, and preferential attachment. Whereas, the second group consists of measures based on ensemble of all paths, such as Katz, Hitting time, Page Rank and SimRank [15, 16]. Irrespective of the similarity metric used, computation of social closeness between any two given users requires the topological structures of the network and/or user profile contents (such as friend list, social interest tags, education, employment details, etc). Briefly, friend recommendations can be performed as follows. (i) Social closeness (referred to as recommendation score) between A and each potential candidate is computed. (ii) The candidates with Top-K scores can be recommended as new friends to A.

The computation of recommendation scores between a target user A and any other user in the given snapshot of a network is straightforward when user's profile information is treated as public. However, with the growing use of OSNs, there have been various concerns about user privacy [17, 18, 19, 20, 21, 22, 23]. Some well-known privacy issues in the social networks are data publishing to a third party, social phishing and analysis on the social data. Due to these privacy concerns, freely

mining the social network data is not allowed or feasible; therefore, researchers from both academia and industry are moving towards developing problem-specific privacy-preserving techniques. What is private to a given user is subjective in nature. In particular to the friend recommendation problem, the friendship between any two users can be treated as sensitive/private information. This assumption is realistic and being supported by many on-line social networks (e.g., Facebook) where users are allowed to hide their friend lists. Most often, when people maintain friendship with trusted ones, there is much information flowing from one to another. Thus, revealing this sensitive information (i.e., friendship) poses a great threat to user privacy through social engineering attacks [24, 25].

Recently, Ratan et al. [26] have conducted a survey by crawling the public profile pages of 1.4 million New York City (NYC) Facebook users in March 2010 and again in June 2011. It was shown that NYC users from their sample have become dramatically private during this period. More specifically, in March 2010 only 17.2% of users in their sample kept their friend lists as private; however, in June 2011, just 15 months later, 52.6% of the users hid their friend lists [26]. This further suggests that users are more concerned about their privacy; therefore, forcing them to make their profile information private. The observation is, in addition to friend list, other contents of a user profile can also be kept private. Nevertheless, without certain personal profile information, identifying new friends is not possible using the existing friend recommendation techniques.

Based on the above discussions, it is clear that there is a strong need to develop privacy-preserving friend recommendation algorithms in OSNs. Along this direction, this work proposes a set of private friend recommendation protocols by assuming different combinations of user's profile information is kept as private.

1.2. ORGANIZATION

The emerging growth of online social networks has opened new doors for various business applications such as promoting a new product across its customers. Besides this, friend recommendation is an important tool for recommending potential candidates as friends to users in order to enhance the development of the entire network structure. Existing friend recommendation methods utilize social network structure and/or user profile information. However, these techniques can no longer be applicable if the privacy of users is taken into consideration. Along this direction, this work proposes four different set of private friend recommendation protocols which are organized into four sections as follows.

First, Section 2 presents an overview of the existing work related to the problem domain. In addition, this section reviews the literature work of secure multiparty computation and highlight the security definition adopted.

Section 3 proposes a two-phase private friend recommendation protocol for recommending friends to a given target user based on the network structure as well as utilizing the real message interaction between users. The proposed protocol computes the recommendation scores of all users who are within a radius of h from the target user in a privacy-preserving manner. This work also addresses some implementation details and point out an inherent security issue in the current online social networks due to the message flow information. To mitigate this issue or to provide better security, this work proposes an extended version of the proposed protocol using randomization technique. In addition, the practical applicability of the proposed approach is discussed extensively through empirical analysis based on different parameters.

Section 4 proposes two private friend recommendation algorithms for users in a social network group G by leveraging both social tags and network topology. For

a given target user u_i, the proposed protocols compute the social closeness scores between u_i and each user in the subset $G_i \subset G$ in a privacy-preserving manner by utilizing an ontology tree T constructed by the domain expert such as the network administrator. The first protocol is more efficient from a user's perspective compared to the second protocol, and this efficiency gain comes at the expense of relaxing the underlying privacy assumptions. On the other hand, the second protocol provides the best security guarantee. In addition, this work empirically analyzes the complexities of the proposed protocols and provides various experimental results.

Section 5 proposes two novel methods to recommend friends for a given target user by using the common neighbors proximity measure in a privacy preserving manner. The first method is based on the properties of an additive homomorphic encryption scheme and also utilizes a universal hash function for efficiency purpose. The second method utilizes the concept of protecting the source privacy through randomizing the message passing path and recommends friends accurately and efficiently. In addition, this work empirically compares the efficiency and accuracy of the proposed protocols, and addresses the implementation details of the two methods in practice. The proposed protocols act as a trade-off among security, accuracy, and efficiency; thus, users can choose between these two protocols depending on the application requirements.

Section 6 considers the scenario of outsourced social networks, where users encrypt their profiles independently and export them to a third-party cloud service provider, such as Google or Amazon. Since the data are encrypted, query processing over encrypted data becomes challenging for the cloud. In particular, this section focuses on the friend recommendation problem over encrypted users' profiles based on the secure k-nearest neighbor (SkNN) technique. More specifically, this work develops two novel SkNN protocols for the cloud to recommend the top k-nearest neighbors as potential friends to a given target user in a privacy-preserving manner. The first

protocol, which acts as a basic solution, leaks some information to the cloud. On the other hand, the second protocol is fully secure, that is, it protects the confidentiality of the data and also hides the data access patterns. However, the second protocol is more expensive compared to the basic protocol. Also, the performance of the proposed protocols under different parameter settings is evaluated.

Finally, this report concludes the contributions of this work and demonstrates several possible directions for future research in Section 7.

2. LITERATURE REVIEW

Social network analyses have been utilized for various business applications [6, 27], such as predicting the future [28] and developing recommender systems [29, 30, 31, 32]. With growing interest of expanding a person's social circle, friend recommendation has become an important service in many online social networks. This section reviews upon the existing friend recommendation algorithms along with the the existing work related to private friend recommendations. Finally, it discusses the literature work on secure multiparty computation along with the security definition adopted in this work.

2.1. EXISTING FRIEND RECOMMENDATION METHODS

Social network analyses have been utilized for various business applications [6, 27], such as predicting the future [28] and developing recommender systems [29, 30, 31, 32]. With growing interest of expanding a person's social circle, friend recommendation has become an important service in many OSNs. Along this direction, researchers from both academia and industry have published much work. In particular, Chen et al. [9] evaluated four recommender algorithms, which utilize social network structure and/or content similarity, in an IBM enterprise social networking site Beehive through personalized surveys. Their analysis showed that algorithms based on social network information produce better-received recommendations. A novel user calibration procedure was proposed by Silva et al. [11] based on a genetic algorithm to optimize the three indices derived from the structural properties of social networks. Xie [12] designed a general friend recommendation framework to recommend friends based on the common interests by characterizing user interests in two dimensions - context (e.g., location and time) and content.

By treating the friend recommendation process as a filtering problem, Naruchitparames et al. [10] developed a two-step approach to provide quality friend recommendations by combining cognitive theory with a Pareto-optimal genetic algorithm. Gou et al. [13] developed a visualization tool (named as SFViz) that allows users to explore for a potential friend with an interest context in social networks. Their method considers both semantic structure in social tags and topological structures in social networks to recommend new friends. The correlation between social and topical features in three popular OSNs: Flickr, Last.fm, and aNobii has been studied by Aiello et al. [33] to analyze friendship prediction. Their results showed that social networks constructed solely from topical similarity captured the actual friendship accurately. Nevertheless, Facebook uses the "People You May Know" feature to recommend friends based on the simple "friend-of-a-friend" approach [34].

2.2. PRIVATE FRIEND RECOMMENDATION (PFR) METHODS

Due to various privacy issues [17, 18, 19, 20, 21, 22, 23], many users keep their profile information as private. Existing friend recommendation techniques [9, 10, 11, 12, 13, 14] do not take users' privacy into consideration; therefore, they cannot be directly applied. Only recently, researchers have focused on developing accurate and efficient PFR methods. Along this direction, Dong et al. [35] proposed a method to securely compute and verify social proximity between two users using cosine similarity in mobile social networks. In their approach, (mobile social network) users physical location is treated as private. Their approach identifies new friends who happen to be in the physical vicinity of the target user. That is, social coordinates (users geographical location) are used to compute the social proximity between users. Nevertheless, their approach assumes that the social coordinates (which may

change often due to the mobility of users) for individual users are pre-computed by a trusted central server which is a violation of user privacy.

Machanavajjhala et al. [36] formally analyzed the trade-offs between accuracy and privacy of private friend recommendations using differential privacy [37]. In their work, the authors used the existing differentially private algorithms as underlying sub-routines and assumed the existence of PFR protocols based on these sub-routines. Also, according to their claims, if privacy is to be preserved when using the common neighbors utility function [15], only users with $\Omega(\log n)$ friends can hope to receive accurate recommendations, where n is the number of users in the graph. Furthermore, the users' privacy in [36] is based on differential privacy. Whereas, in this work, privacy guarantees are based on an entirely different security model, namely the semi-honest security definitions from the field of secure multiparty computation (SMC) [38, 39]. Under the SMC model, this work develops accurate and private friend recommendation protocols.

2.3. SECURE MULTIPARTY COMPUTATION AND ITS SECURITY DEFINITION

Due to the growing concerns about privacy and the distributed nature of data, secure multiparty computation (SMC) plays an important role in solving a wide-range of applications. Some of these applications include secure electronic voting [40], private auctioning and bidding [41, 42], and privacy-preserving data mining [43, 44].

Consider a scenario where multiple parties, each with their private input a_i, wish to collaborate and compute a common functionality f by preserving the privacy of each user. To achieve that, parties have to exchange messages and perform some local computations until all the parties get the desired output. In the literature, this is referred to as secure multiparty computation (SMC). More formally, SMC is

the evaluation of the function $f(a_1, \ldots, a_n) = (b_1, \ldots, b_n)$ such that the output b_i is known to party P_i and the input a_i of each party is kept private. This definition was first introduced by Yao to solve the famous Millionaires' problem in 1982 [38, 39]. Let Alice and Bob be two millionares with their respective wealth a and b. The goal of Alice and Bob is to determine who is richer without revealing their wealth to one another. More precisely, the functionality one need to evaluate is "greater than", that is, whether a is greater than b or not. The first general and provably secure solution, for a two-party case, was developed by Yao and it was also demonstrated that any function that can be described by a polynomial size boolean circuit of logarithm depth can be solved securely [38, 39]. This work was extended to multiparty computations by Goldreich et al. [45]. It was proved in [45] that any computation which can be done in polynomial time by a single party can also be done securely by multiple parties. Since then much work has been published for the multiparty case [46, 47, 48, 49].

In this work, privacy/security is closely related to the amount of information disclosed during the execution of a protocol. There are many ways to define information disclosure. To maximize privacy or minimize information disclosure, this work adopts the security definitions in the literature of SMC [49, 50]. There are two common adversarial models under SMC: semi-honest and malicious. An adversarial model generally specifies what an adversary or attacker is allowed to do during an execution for a security protocol. In the semi-honest model, an attacker (i.e., one of the participating parties) is expected to follow the prescribed steps of a protocol. However, the attacker can compute any additional information based on his or her private input, output and messages received during an execution of a secure protocol. As a result, whatever can be inferred from the private input and output of an attacker is not considered as a privacy violation. An adversary in the semi-honest model can be treated as a passive attacker; on the other hand, an adversary in the

malicious model can be treated as an active attacker who can arbitrarily diverge from the normal execution of a protocol.

In this dissertation, to develop secure and efficient protocols, all the participating parties are assumed to be semi-honest. Detailed security definitions and models can be found in [49, 50]. The following definition captures the above discussion regarding a secure protocol under the semi-honest model.

Definition 1. *Let a_i be the input of party P_i, $\prod_i(\pi)$ be P_i's execution image of the protocol π and b_i be the result computed from π for party P_i. π is secure if $\prod_i(\pi)$ can be simulated from $\langle a_i, b_i \rangle$ and distribution of the simulated image is computationally indistinguishable from $\prod_i(\pi)$.*

In the above definition, an execution image generally includes the input, the output and the messages communicated during an execution of a protocol. Briefly, to prove a protocol is secure, it is required to show that the execution image of a protocol does not leak any information regarding the private inputs of participating parties [50].

3. STRUCTURAL AND MESSAGE BASED PRIVATE FRIEND RECOMMENDATION

In general, a recommendation score between any two given users can be computed based on the network topology and/or user profile contents (such as previous employer, location and hobbies). For the past few years, researchers have been focused on developing hybrid friend recommendation algorithms [9, 13] to take advantage of both approaches. Recently, Bi-Ru Dai et al. [14] proposed a new friend recommendation algorithm (denoted as CSM - meaning "Combine Structure and Messages") by utilizing the real messages communicated between the users as well as the network structure. To be concrete, this work computes the recommendation scores between users based on the similarity metric given in [14]. More details are given in the later part of this section.

The computation of recommendation scores based on the similarity metric given in [14] is straight-forward if user's data is public. However, as users are more concerned about their privacy [19, 21, 22, 23, 51], many online social networks have provided various privacy settings for users to keep their data private. In general, users are allowed to keep their friend lists, profile information etc., as private information. Under this scenario, the computation of recommendation scores is non-trivial. This work proposes a two-phase private friend recommendation algorithm based on the similarity metric proposed in [14]. The proposed method computes the recommendation scores between A and all potential users who are h-hop away from A in a privacy-preserving manner. Figure 3.1, shows a sample network for target user *Lee* with $h = 3$. In practice, as proposed by Stanley Milgram [52], any two people can get acquainted each other through six degree of separation (i.e., $1 < h \leq 6$) in the network.

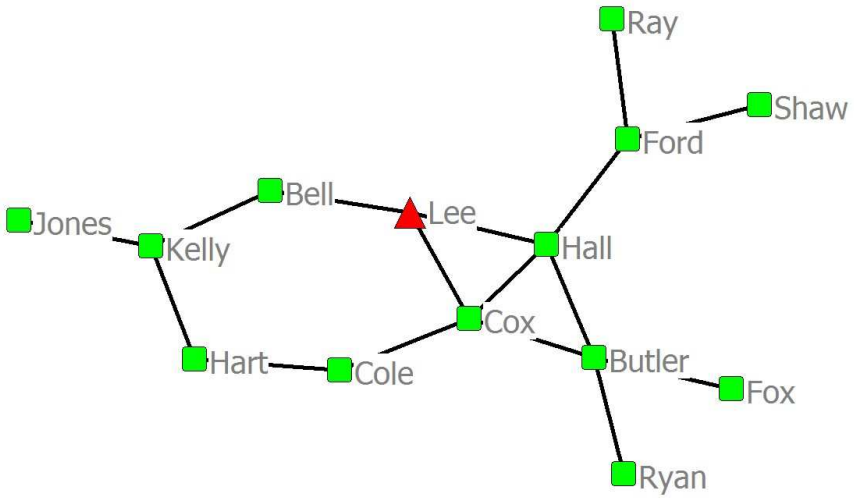


Figure 3.1: A sample social network for *Lee* with $h = 3$

Also, this work discusses various practical implementation details of the proposed protocol. In addition, this work points out an inherent security issue in the current online social networks due to the message flow information among various users. To mitigate this issue or to provide better security, an extension to the proposed protocol is developed using randomization technique.

3.1. PROBLEM DEFINITION

Consider a social network graph G_s with the nodes denoting the users and the (directed) weighted edge between any two nodes denoting the number of real message interactions between them. Since the message interaction can be bi-directional, one can take the minimum number of messages, as mentioned in [14, 53], as the actual weight of the edge (denoting the strength of the relationship). A sample minimum message interaction between various users (for $h = 3$) in *Lee*'s network is as shown in Figure 3.2. In general, if user A sends n_1 messages to B and B sends n_2 messages to A, then the weight of the edge between A and B is taken as $\min(n_1, n_2)$. This further

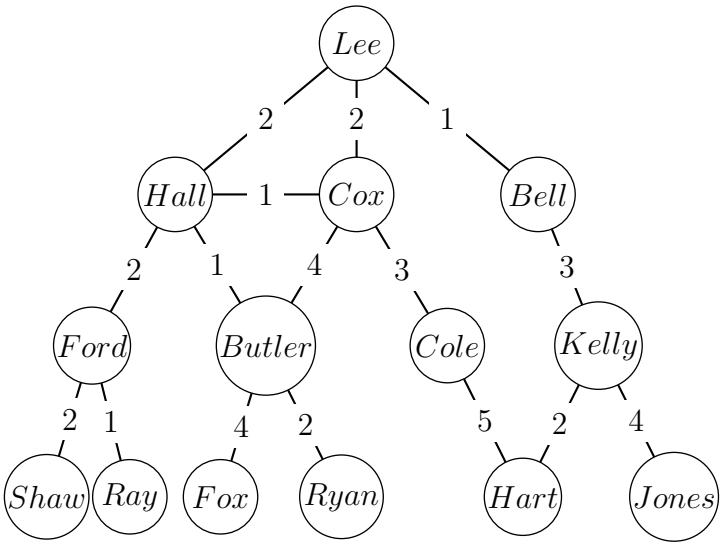


Figure 3.2: Message interaction between different users in *Lee*'s network

implies that the weight of the edge between any two friends is directly correlated to the strength of their relationship (i.e., larger weight indicates stronger friendship).

For a target user A (i.e., a user who wants to make new friends), generate a candidate network with A as the root and an edge between the users denoting the number (minimum) of real message interactions. Note that the users who are 1-hop away from A are actually his/her friends. In order to generate the candidate network, one has to remove the links between users at the same level. E.g., refer to Figure 3.2, one can generate the candidate network by removing the link between $Hall$ and Cox (since they are on the same level). The recommendation score (RS) between A and a user U who is l-hop ($2 \leq l \leq h$) away from A in the candidate network is given as [14]:

$$RS(A,U) = \left(\sum_{k} \left(|P_k(A,U)| * \prod_{i} C(S_{i-1}, S_i) \right) \right) * \frac{D_U}{TN} \tag{3.1}$$

where $P_k(A,U)$ denotes all the intermediate users on the k^{th} shortest path starting from A (root) to user U, $|P_k(A,U)|$ is the total number of messages along path $P_k(A,U)$, and let $L(i)$ be the set of all users at level i (i.e., i-hop away from A).

$S_i \in P_k(A,U) \cap L(i)$, for $i = 1, \ldots, l-1$, where $U \in L(l)$. Note that S_0 denotes the root user A. $C(S_{i-1}, S_i)$ denotes the proportion of messages between users S_i and S_{i-1} to the total number of messages at level i. Here, user S_{i-1} is the parent of user S_i in the corresponding candidate network; D_U denotes the degree of U and TN denotes the total number of users in the candidate network.

When the privacy of users is taken into consideration, the computation of above mentioned recommendation score is not straight-forward. More specifically, this work assumes the following private information (PI) for user U:

(i). **PI 1 - Friendship:** The friendship between any two users U and V is not revealed to any other user.

(ii). **PI 2 - Strength of Friendship:** The weight of an edge between U and V, denoted as $C_{U,V}$, is not revealed to users other than U and V.

(iii). **PI 3 - Degree:** The size of the friend list of U is not revealed to other users.

Without loss of generality, let $U_1, \ldots, U_n$ be the set of potential candidates who are at most l-hop ($2 \leq l \leq h$) away from A. The goal of this work is to develop a private friend recommendation (PFR) protocol which is formally defined as follows:

$$\text{PFR}(A, F(A), U_1, \ldots, U_n) \rightarrow \Gamma \tag{3.2}$$

where $F(A)$ denote the friend list of user A. Γ is defined as:

$$\Gamma = \{\langle \widetilde{RS}(A, U_1), U_1 \rangle, \ldots, \langle \widetilde{RS}(A, U_n), U_n \rangle\}$$

Here, $\widetilde{RS}(A, U_j)$ is the new recommendation score for U_j which is correlated to the actual score $RS(A, U_j)$ (based on Equation 3.1) as below, for $1 \leq j \leq n$:

$$\widetilde{RS}(A, U_j) \quad = \quad M_h * TN * RS(A, U_j)$$

M_h is the normalizing factor for a user at h-hop away from A and TN is the number of users in the candidate network. For any fixed h and A, the observation is that M_h and TN are constants. At the end of the PFR protocol, the values of $\widetilde{RS}(A, U_j)$ and U_j, for $1 \leq j \leq n$, are known only to A and the privacy of each user (PI 1, 2, and 3) is preserved. In practice, since the friend lists can be large, the number of scores returned to A can be in the hundreds. Therefore, a more effective way is to simply select Top-K users as the final set of friend recommendations.

3.2. MAIN CONTRIBUTIONS

The proposed protocol computes the recommendation scores between a target user A and all potential candidates who are at most l-hop ($2 \leq l \leq h$) away from A in a privacy-preserving manner. The main contributions of this particular work are summarized below:

- **Security -** The proposed protocol guarantees that the friend lists, the strength of friendships, and the friend list sizes of each user are kept as private from other users. However, this work identifies an inherent security issue that may leak valuable information to the network administrator in the current online social networks which is also applicable to the proposed protocol. To mitigate this risk, this work also proposes an extended version of the proposed protocol using randomization technique.
- **Accuracy -** The proposed protocols compute the recommendation scores which are scaled by a constant factor $M_h * TN$; therefore, the relative ordering among the scores is preserved. Hence, the proposed protocols guarantee the same kind of effectiveness similar to the CSM method [14]. That is, the final Top-K list of recommended users in the proposed protocols is the same as that in [14] and is independent of K.

- **Efficiency -** In the empirical analysis, this work shows the practical value of PFR through various experiments. Also, it shows that the efficiency of the extended version is very close to that of PFR. The experiments show that the computation costs incurred on the internal users in the proposed protocols are very small; therefore, the proposed protocols are very efficient from an internal user's perspective.

3.3. RELATED WORK

As mentioned earlier, friend recommendation is a very useful application for both users and the social network provider. Through social recommendations, users are allowed to make new friends; therefore, expanding their social connections. In addition, it helps the social network provider in a way to enhance the development of entire network structure.

3.3.1. Existing Friend Recommendation Methods. In general, recommendation scores between any two given users can be computed either based on the network topology [10, 11] and/or user profile contents [12].

Only recently, researchers have focused on developing hybrid friend recommendation algorithms [9, 13] to take advantages of both approaches. As an independent work, Lo et al. [53] proposed a graph-based friend recommendation algorithm using a weighted minimum-message ratio as the scoring metric. This work was later improved in [14] by taking the evolution of entire network into consideration. The computation of recommendation scores based on the metric given in [14] is straightforward when users data is public. However, due to the growing concerns over user privacy [19, 21, 22, 23, 51], many users prefer to keep their profile data (including their friend lists) as private. Along this direction, many online social networks such as Facebook, provide various privacy settings for users to make their data private.

Therefore, the above existing methods are not applicable if privacy of users is taken into consideration.

To make friend recommendations possible even in privacy-sensitive environments, this work proposes a two-phase PFR protocol based on the similarity metric given in [14]. Furthermore, this work addresses an inherent security issue in the current online social networks. To overcome this issue, an extended version to the proposed PFR protocol is also proposed.

3.3.2. Existing PFR Protocols. In the literature, that there has not been much work done in developing efficient PFR protocols based on different metrics. Dong et al. [35] proposed a method to securely compute the social proximity between users in a mobile social network. They have used the cosine similarity metric to compute how close two give users are by treating user's location as private.

Machanavajjhala et al. [36] analyzed the trade-offs between accuracy and privacy for private friend recommendation algorithms based on differential privacy [37, 54]. The work in this section is entirely different from theirs since the security guarantee in this work is based on the the well-known semi-honest security definition of secure multiparty computation (SMC) [38, 39, 50]. In addition, they use a different similarity metric, namely common neighbors [15] whereas this work is based on the scoring metric given in Equation 3.1.

In general, different metrics have different advantages. Secure protocols designed for one metric often may not work for other metrics. Therefore, there is a strong need to develop a secure protocol based on particular scoring function. In particular, this work focuses on developing a secure PFR protocol based on the scoring function given in Equation 3.1. Table 3.1 presents some common notations used extensively in this section.

Table 3.1: Common notations used in PFR

HPEnc$^+$	An Additive Homomorphic Probabilistic Encryption System
T	A trusted party (such as network administrator)
$\langle E, D\rangle$	A pair of HPEnc$^+$ based encryption and decryption functions
$\langle pk, pr\rangle$	A public and private key pair corresponding to $\langle E, D\rangle$
$F(U)$	Friend list of user U
m	Size of friend list of target user A
$C_{U,V}$	Minimum number of messages exchanged between U and V (or weight of edge between U and V)
$L(i)$	List of all users at level i in the corresponding candidate network
$M_{i-1,i}$	Minimum number of messages exchanged between users at $L(i-1)$ and $L(i)$
$C(S_{i-1}, S_i)$	Ratio of C_{S_{i-1},S_i} to $M_{i-1,i}$
M_l, M'_l	Normalization and Scalar factors for a user $\in L(l)$

3.4. ORDER PRESERVING SCORING FUNCTION

The original scoring function [14] given in Equation 3.1 contains a rational factor (i.e., $C(S_{i-1}, S_i)$) which varies with i, for $1 \leq i \leq l-1$ and $2 \leq l \leq h$. Therefore, to perform encryption operations, this work defines a new scoring function (producing an integer value) based on Equation 3.1 such that the relative rankings among the final recommendation scores are preserved.

3.4.1. Normalization Factor. Given a snapshot of the network for A, the normalization factor for a user l-hop (or friend) away from A (where $2 \leq l \leq h$) is defined as:

$$M_l = \prod_{i=1}^{l-1} M_{i-1,i} \tag{3.3}$$

where $M_{i-1,i}$, denoting the total number of messages exchanged between users at $L(i-1)$ and $L(i)$, is as given below.

$$M_{i-1,i} = \sum_{\substack{U \in L(i-1) \\ V \in L(i)}} C_{U,V}$$

$C_{U,V}$ denotes the minimum number of messages exchanged between users U and V. This work explicitly assumes $M_1 = 1$ since users who are 1-hop from A are already

friends of A. For any two potential candidates who are l-hop away from A, the observation is that the two candidates have the same normalization factor.

Example 1. *Refer to Figure 3.2. Consider the potential candidate Cole who is 2 hops away from Lee. Here,* $L(0) = \langle Lee \rangle$ *and* $L(1) = \langle Hall, Cox, Bell \rangle$*. The normalization factor for Cole is* $M_2 = M_{0,1} = C_{Lee,Hall} + C_{Lee,Cox} + C_{Lee,Bell} = 5$*. Note that the normalization factor for* $Ford, Butler$*, and* $Kelly$ *(who are also 2 hops away from Lee) is the same as Cole. Similarly, it is clear that* $M_{1,2} = 13$*. By substituting these values in Equation 3.3, the normalization factor for users at level 3 is* $M_3 = \prod_{i=1}^{2} M_{i-1,i} = M_{0,1} * M_{1,2} = 65$*.*

Observation 1. *For any user* $S_{i-1} \in L(i-1)$ *and* $S_i \in L(i)$*, one can observe that the value of* $C(S_{i-1}, S_i)$ *is equivalent to* $\frac{C_{S_{i-1},S_i}}{M_{i-1,i}}$*. Therefore, for a potential user* U *at level* l*, the rational factor in Equation 3.1 can be simplified as follows:*

$$\prod_{i=1}^{l-1} C(S_{i-1}, S_i) = \prod_{i=1}^{l-1} \frac{C_{S_{i-1},S_i}}{M_{i-1,i}} = \frac{1}{M_l} \prod_{i=1}^{l-1} C_{S_{i-1},S_i}$$

3.4.2. Scalar Factor. Given a target user A and h, the scalar factor for a user at level l, for $1 \leq l \leq h$, can be defined as follows:

$$M'_l = \frac{M_h}{M_l} = \frac{M_{0,1} * \ldots * M_{h-2,h-1}}{M_{0,1} * \ldots * M_{l-2,l-1}} \tag{3.4}$$

where M_l is the normalization factor for a user belonging to $L(l)$. In addition, the observation is that M'_l is the same for all users who are at same level l. Furthermore, when $l = h$, it is clear that $M'_h = 1$. This further implies that $M'_1 = M_h$. From Figure 3.2, the scalar factor for $Cole$ is $M'_2 = \frac{M_3}{M_2} = M_{1,2} = 13$.

Definition 2. *For any given target user* A *and potential candidate* U *who is* l *hops away from* A*, the new scoring function (denoted as* $\widetilde{RS}(A,U)$*) is defined as follows:*

$$\widetilde{RS}(A,U) = M'_l * \left(\sum_k \left(|P_k(A,U)| * \prod_i C_{S_{i-1},S_i} \right) \right) * D_U \tag{3.5}$$

Note that C_{S_{i-1},S_i} is the weight of edge between parent user S_{i-1} and its child user S_i on the k^{th} shortest path from A to U, for $1 \leq i \leq l-1$. Based on Equations 3.3 and 3.4, and by using Observation 1, one can re-write Equation 3.5 as below.

$$\begin{aligned}
\widetilde{RS}(A,U) &= \frac{M_h}{M_l} * \left(\sum_k \left(|P_k(A,U)| * \prod_i C_{S_{i-1},S_i} \right) \right) * D_U \\
&= M_h * \left(\sum_k \left(|P_k(A,U)| * \prod_i \frac{C_{S_{i-1},S_i}}{M_{i-1,i}} \right) \right) * D_U \\
&= M_h * TN * \left(\sum_k \left(|P_k(A,U)| * \prod_i C(S_{i-1},S_i) \right) \right) * \frac{D_U}{TN} \\
&= M_h * TN * RS(A,U)
\end{aligned}$$

The values of M_h and TN are constants for any given snapshot of the social network (for a fixed h). Therefore, the relative orderings among the recommendation scores of the potential candidates based on Equation 3.5 are preserved. That is, for any two potential users U and V if $RS(A,U) > RS(A,V)$, then the new scoring function guarantees that $\widetilde{RS}(A,U) > \widetilde{RS}(A,V)$ for any fixed h and A, and vice versa.

3.4.3. Computation of Recommendation Score. Refer to Figure 3.2 and let us consider the case of computing the recommendation score between Lee and Fox. Here, Fox has two shortest paths from Lee; $P_1(Lee, Fox) = \{Lee, Hall, Butler, Fox\}$ and $P_2(Lee, Fox) = \{Lee, Cox, Butler, Fox\}$. The total (minimum) number of messages along the first path i.e., $|P_1(Lee, Fox)|$ is 7. Similarly, $|P_2(Lee, Fox)| = 10$. Along $P_1(Lee, Fox)$, there exist two internal users $Hall$ and $Butler$ who are respectively 1 and 2 hops away from Lee. In addition, $C_{Lee,Hall} = 2$ and $C_{Hall,Butler} = 1$. Similarly, for the path $P_2(Lee, Fox)$, $C_{Lee,Cox} = 2$ and $C_{Cox,Butler} = 4$. Since Fox is 3 hops away from Lee, her scaling factor M'_3 is 1. By substituting the above values

in Equation 3.5, the recommendation score for Fox is given as:

$$\widetilde{RS}(Lee, Fox) = 1 * [7 * 2 * 1 + 10 * 2 * 4] * D_{Fox} = 94 * D_{Fox}$$

Whereas, the actual recommendation score for Fox, following from Equation 3.1, is given by:

$$\begin{aligned} RS(Lee, Fox) &= \left[7 * \frac{2}{5} * \frac{1}{13} + 10 * \frac{2}{5} * \frac{4}{13}\right] * \frac{D_{Fox}}{TN} \\ &= \frac{1}{65} * 94 * \frac{D_{Fox}}{TN} \end{aligned}$$

where D_{Fox} is the degree (size of friend list) of Fox and TN denotes the size of the candidate network. It is clear that $\widetilde{RS}(Lee, Fox) = M_h * TN * RS(Lee, Fox)$, where $M_h = M_{0,1} * M_{1,2} = 5 * 13 = 65$.

3.5. THE PROPOSED PFR PROTOCOL

This sub-section presents the proposed private friend recommendation (termed as PFR) protocol which computes the recommendation scores between the target user A and all potential candidates who are at most h-hop (> 1) away from A based on Equation 3.5. This work explicitly considers the following assumptions:

1. If $U \in F(V)$, then $V \in F(U)$, and $C_{U,V}$ is known only to U and V. Also, let $F(A) = \langle B_1, \ldots, B_m \rangle$ denote the friend list of A.
2. Each user has a unique user ID (for example, Facebook user ID is generally at most 128-bit integer).
3. There exists a third party T (e.g., network administrator) who generates a pair of encryption and decryption function $\langle E, D \rangle$ for A based on the additive homomorphic probabilistic encryption scheme (HPEnc$^+$) such as the Paillier

cryptosystem [55]. The corresponding private key pr is known only to T and the public key pk is public. In addition, let N be the group size (usually of 1024 bits). For any two given plaintexts $m_1, m_2 \in \mathbb{Z}_N$, the HPEnc$^+$ system exhibits the following properties:

(a) **Homomorphic Addition:** $E_{pk}(m_1+m_2) \leftarrow E_{pk}(m_1)*E_{pk}(m_2) \bmod N^2$;

(b) **Homomorphic Multiplication:** $E_{pk}(m_1 * m_2) \leftarrow E(m_2)^{m_1} \bmod N^2$;

(c) **Semantic Security:** The encryption scheme is semantically secure as defined in [49, 56]. Briefly, given a set of ciphertexts, an adversary cannot deduce any additional information about the plaintext.

To generate the candidate network, one need to omit the messages between users who are at the same level. For example, in Figure 3.2, one should not consider $C_{Hall,Cox}$ for computing the recommendation scores in the PFR protocol (as mentioned in [14, 53]). Thus, to explicitly generate the candidate network, this work includes an initialization step as follows. Initially, A generates a counter $t = h - 1$ and passes it over to his/her friends. Upon receiving the counter, each intermediate user U stores the value of received counter (locally) and also stores the parent user who sent the counter to U (denoted as $Pr(U)$). After this, U decrements the counter by 1 and sends it to his/her friends. This process continues until users at h-hop from A receive a counter of $t = 0$. Since a user can receive multiple counter values, the following observations are considered.

Observation 2. *Consider user U, who is l-hop away from A and $1 \leq l \leq h$, receiving multiple t values. This work addresses the following two cases:*

Case 1: If the counter values are same, then U has multiple shortest paths (with parents of U on the same level). In this case, U considers one of the parents (can be chosen randomly) as actual parent $Pr(U)$ and any further communication

happens only with that parent. E.g., refer to Figure 3.2, "$Hart$" receives $t = 0$ from both $Cole$ and $Kelly$. Therefore, he can pick one of them, say $Kelly$, as $Pr(U)$.

Case 2: If U receives different values of t which happens when U receives counters from parents who are at different levels. In this case, U selects one of the parent user who sent the maximum t value as $Pr(U)$. In the PFR protocol, the child users of U (denoted as $Ch(U)$) are users belonging to $F(U) - R(U)$, where $R(U)$ denotes the set of users who have sent a counter value to U. The important observation here is U omits the messages exchanged with the users who have sent smaller counter values (also dumps the corresponding counter). This further implies that, U considers only messages exchanged between him/her and either $Pr(U)$ or $Ch(U)$ (therefore forming a candidate network by omitting messages with users on the same level). An example to this case is user "Cox" (refer to Figure 3.2). Here, Cox receives $t = 2$ and $t = 1$ from Lee and $Hall$ respectively. Therefore, Cox treats Lee as the actual parent user and omits $C_{Cox,Hall}$.

At the end of the initialization step, based on Observation 2, each internal user U who is l-hop away from A, for $1 \leq l \leq h$, has the values of $t, pk, Pr(U)$ and $Ch(U)$. Apart from the above initialization step, the proposed PFR protocol mainly consists of the following two phases:

Phase 1 - Secure Computation of Scalar Factors: During Phase 1, A computes the list of encrypted scalar factors (denoted as Φ, where Φ_{l-1} denotes the encrypted scalar factor for level l and $2 \leq l \leq h$) in a privacy-preserving manner. This phase utilizes a secure multiplication protocol (only if $h > 3$) as a building block. At the end, only A knows Φ and nothing is revealed to other users.

Phase 2 - Secure Computation of Recommendation Scores: Following from Phase 1, A (with Φ as input), T and other internal users jointly compute the recommendation scores of all potential candidates who are l-hop away from A, for $2 \leq l \leq h$. This phase utilizes a secure multiplication and addition protocol as a

building block. The final recommendation scores and the corresponding user IDs are revealed only to A and nothing is revealed to other users.

To start with, A chooses the value of h^* and executes the initialization step as explained earlier. Then, during Phase 1, A decides whether there is a need to take the help of other users in order to generate Φ. If $h = 2$, A computes Φ locally. Otherwise, for $h > 2$, A computes Φ with the help of internal users. After this, during Phase 2, A sends necessary information to B_i along with his/her user ID and Φ, for $1 \leq i \leq m$. Then, each intermediate user U_j receives the necessary information from $Pr(U_j)$, generates his/her encrypted partial scores (only if U_j is not already a friend of A) and sends the encrypted partial scores to A. In addition, if the value of t (stored during initialization step) of U_j is greater than 0, he/she computes the necessary information (for $t > 0$) and sends it to his/her corresponding child friends. After receiving all the encrypted partial scores, A and T involve in a secure multiplication and addition protocol to compute the recommendation scores for each potential candidate U_j. At the end of this step, only A knows the user IDs of all potential friends along with their recommendation scores (computed based on Equation 3.5). The main steps of PFR are shown in Algorithm 1. Now, the steps involved in each of the two phases are discussed in detail.

3.5.1. Phase 1 - Secure Computation of Scalar Factors. If the value of h is 2, then only the child friends of A's friends are considered as the potential candidates. Since the scalar factor for users at $l = 2$ is $M'_2 = 1$, A simply sets $\Phi_1 = E_{pk}(1)$ for security reasons. When $h > 2$, A does not have necessary information to compute the encryption of scalar factors (such as M'_3) since the potential candidates can belong to any $L(l)$, where $2 \leq l \leq h$. Therefore, when $h > 2$, A computes Φ, with the help of internal users who are at most $h - 2$ hops away from A. Note that the potential candidates who are at most $h - 2$ hops away from A are sufficient to

*Note that h should always be greater than 1. Because, if $h = 1$, then $l = 1$ implies the potential candidates who are 1-hop away from A who are already friends of A.

generate the encryptions of all scalar factors because the partial scores of $M_{h-2,h-1}$ are known to users belonging to $L(h-2)$. Note that, irrespective of the value of h, $\Phi_{h-1} = E_{pk}(1)$ always hold. Phase 1 involves steps 1 to 16 as shown in Algorithm 1.

In order to compute Φ, for $h > 2$, A simply waits for internal users with $t \geq 2$ to send in the aggregated data. To start with, each internal user U_j (including B_i) performs the following operations based on his/her counter value t:

1. Compute $X_{U_j} = E_{pk}(\sum_{i=1}^{s} C_{U_j,V_i})$, where V_i is the child friend of U_j and $s = |Ch(U_j)|$
2. Create a vector L_{U_j} of size $t-1$; sets $L_{U_j}[t-1]$ to X_{U_j}
3. If $t > 2$, U_j receives L_{V_i} from V_i and updates L_{U_j} by aggregating L_{V_i} component-wise as follows, and sends it to $Pr(U_j)$.

$$L_{U_j}[k] = \prod_{i=1}^{s} L_{V_i}[k] \bmod N^2, \text{ for } 1 \leq k \leq t-2$$

The above process forwards the aggregated data at each internal user in a bottom-up fashion. At the end, A receives L_{B_i} from B_i, for $1 \leq i \leq m$. After this, A generates the final aggregated encrypted list (L_A) and proceeds as follows:

1. $L_A[k] = \prod_{i=1}^{m} L_{B_i}[k] \bmod N^2$, for $1 \leq k \leq |L_{B_i}|$, where L_{B_i} denote the aggregated list received from B_i. The observation is $|L_{B_i}| = h-2$, for $1 \leq i \leq m$.
2. Assign the encrypted scalar factor for level h as $\Phi_{h-1} = E_{pk}(1)$. If $h = 3$, set $\Phi_1 \leftarrow L_A[1]$. Else, let $L_A = \langle E_{pk}(x_1), \ldots, E_{pk}(x_{h-2}) \rangle$. Using secure multiplication (SMP) given in Algorithm 2, A and T compute Φ from L_A as below.

$$\Phi_l \leftarrow E_{pk}\left(\prod_{j=1}^{h-l-1} x_j\right), \text{for } 1 \leq l \leq h-2$$

The SMP protocol is one of the basic building blocks in the field of secure multiparty computation (SMC) [50]. The basic concept of the SMP protocol is based on the

Algorithm 1 PFR

Require: pr is private to T, h is private to A, U_j knows $\langle t, Pr(U_j), Ch(U_j)\rangle$

{**Steps 1 - 7 performed by** U_j **with** $t \geq 2$}

$s \leftarrow |Ch(U_j)|$

$X_{U_j} \leftarrow E_{pk}(\sum_{i=1}^{s} C_{U_j,V_i})$, where $V_i \in Ch(U_j)$

$L_{U_j}[t-1] \leftarrow X_{U_j}$

if $t > 2$ and U_j received L_{V_i} from V_i **then**

  $L_{U_j}[k] \leftarrow \prod_{i=1}^{s} L_{V_i}[k] \bmod N^2$, for $1 \leq k \leq t-2$

end if

send L_{U_j} to $Pr(U_j)$

{**Steps 8 - 16 performed by** A **and** T}

$\Phi_{h-1} = E_{pk}(1)$

if $h \geq 3$ **then**

  $L_A[k] \leftarrow \prod_{i=1}^{m} L_{B_i}[k] \bmod N^2$, for $1 \leq k \leq h-2$

  if $h = 3$ **then**

    $\Phi_1 \leftarrow L_A$

  else

    Compute Φ using L_A as input to the SMP protocol

  end if

end if

{**Steps 17 - 21 performed by** A}

for all $B_i \in Ch(A)$ **do**

  $\alpha_1 \leftarrow E_{pk}(C_{A,B_i})$

  $\alpha_l \leftarrow \Phi_{l-1}^{C_{A,B_i}} \bmod N^2$, for $2 \leq l \leq h$

  send A, Φ, and α to B_i (note that α is different for each B_i)

end for

{**Steps 22 - 36 performed by** U_j}

receive A, Φ, and α from $Y = Pr(U_j)$

if $A \in F(U_j)$ **then**

  send A, Φ and α to each $V_i \in Ch(U_j)$

else

  compute $\beta_j \leftarrow \alpha_1^{D_{U_j}} \bmod N^2$

  compute $\gamma_j \leftarrow \alpha_2 * \Phi_1^{C_{Y,U_j}} \bmod N^2$

  $Z_j \leftarrow \{E_{pk}(U_j), \langle \beta_j, \gamma_j \rangle\}$

  send Z_j to A

end if

if $t > 0$ **then**

  $\Phi_l \leftarrow \Phi_{l+1}$, for $1 \leq l \leq t$

  $\alpha_1 \leftarrow \alpha_1^{C_{Y,U_j}} \bmod N^2$

  $\alpha_l \leftarrow \alpha_{l+1} * \Phi_{l-1}^{C_{Y,U_j}} \bmod N^2$, for $2 \leq l \leq t+1$

  send A, Φ and α to each $V_i \in Ch(U_j)$

end if

$(\widetilde{RS}(A, U_j), U_j) \leftarrow \text{SMPA}(Z_j)$, for each Z_j

following property which holds for any given $a, b \in \mathbb{Z}_N$:

$$a * b = (a + r_1) * (b + r_2) - a * r_2 - b * r_1 - r_1 * r_2 \tag{3.6}$$

where all the arithmetic operations are performed under $\mathbb{Z}_N$. Given that A has input $E_{pk}(a)$ and $E_{pk}(b)$, the SMP protocol computes $E_{pk}(a * b)$ as the output (which will be revealed only to A) without disclosing the values of a and b to either A or T. The output of Phase 1 is the list of encrypted scalar factors (in order) for each level. More specifically,

$$\Phi_l = E_{pk}(M'_{l+1}), \text{ for } 1 \leq l \leq h-1$$

where M'_{l+1} is the scalar factor for users at $(l+1)$-hop away from A. If the maximum value of h is 6 (sufficient for most situations), the maximum size of L_A is 4. Therefore, Phase 1 is bounded by 2 instantiations of the SMP Protocol.

Theorem 1. *The output of Phase 1 is the list of encrypted scalar factors (in order) for each level. That is, Φ_l is equivalent to the encryption of scalar factor for users at level $l+1$, where $1 \leq l \leq h-1$. Formally,*

$$\Phi_l = E_{pk}(M'_{l+1})$$

where M'_{l+1} is the scalar factor for users at $l+1$ hops away from A.

Proof. For $h = 2$, since $M'_2 = 1$, it is clear that $\Phi_1 = E_{pk}(1) = E_{pk}(M'_2)$. Note that irrespective of the value of h, $\Phi_{h-1} = E_{pk}(1) = E_{pk}(M'_h)$ always holds. When $h \geq 3$, initially the internal user X with $t = 2$ (denoting level $h-2$) sends $L_X = E_{pk}(\sum_{i=1}^{|Ch(X)|} C_{X,Y_i})$ to Z, where $Y_i \in Ch(X)$ and $Z = Pr(X)$. Then, Z aggregates the data received from $Ch(Z)$. Without loss of generality, let Z receives $L_{X_1}, \ldots, L_{X_d}$, where $X_i \in Ch(Z)$. Then, the aggregated entry in L_Z is $L_Z[1] = L_{X_1}[1] * \ldots * L_{X_d}[1]$. In addition, Z sets $L_Z[2] = E_{pk}(\sum_{i=1}^{|Ch(Z)|} C_{Z,X_i})$. Since the data are aggregated

Algorithm 2 SMP$(E_{pk}(a), E_{pk}(b)) \rightarrow E_{pk}(a * b)$

Require: A has $E_{pk}(a)$ and $E_{pk}(b)$

1: A:

(a). Pick two random numbers $r_a, r_b \in \mathbb{Z}_N$

(b). $z_a \leftarrow E_{pk}(a) * E_{pk}(r_a) \bmod N^2$

(c). $z_b \leftarrow E_{pk}(b) * E_{pk}(r_b) \bmod N^2$; send z_a, z_b to T

2: T:

(a). Receive z_a and z_b from A

(b). $u_a \leftarrow D_{pr}(z_a)$; $u_b \leftarrow D_{pr}(z_b)$

(c). Compute $u = u_a * u_b \bmod N$

(d). $v \leftarrow E_{pk}(u)$; send v to A

3: A:

(a). Receive v from T

(b). $s \leftarrow v * E_{pk}(a)^{N-r_b} \bmod N^2$

(c). $s' \leftarrow s * E_{pk}(b)^{N-r_a} \bmod N^2$

(d). $E_{pk}(a * b) \leftarrow s' * E_{pk}(r_a * r_b)^{N-1} \bmod N^2$

component-wise, l^{th} component in L_Z is equivalent to the encryption of summation of (minimum) number of messages exchanged between users at $L(h-l-1)$ and $L(h-l)$ under sub-tree of Z. (Note that, following from Observation 2, if X_i has multiple parents, then he/she will send L_{X_i} to only actual parent user $Pr(X_i)$). This aggregation process continues at each level in a bottom-up fashion. Finally, when A computes L_A (by aggregating the L_{B_i}'s component-wise, for $1 \leq i \leq m$), the l^{th} component in L_A is equivalent to the encryption of sum of (minimum) number of messages exchanged between users at $L(h-l-1)$ and $L(h-l)$, that is, $L_A[l] = E_{pk}(M_{h-l-1,h-l})$, for $1 \leq l \leq h-2$. As mentioned earlier, let $L_A = \langle E_{pk}(x_1), \ldots, E_{pk}(x_{h-2}) \rangle$, where $x_l = M_{h-l-1,h-l}$, for $1 \leq l \leq h-2$. Based on the above discussions, consider the following two scenarios:

Scenario 1: When $h = 3$, it is clear that $|L_A| = 1$ and Φ_1 gives the encrypted scalar factor for users at level 2 as shown below.

$$\begin{aligned}
\Phi_1 &= L_A[1] \\
&= E_{pk}(M_{1,2}) \\
&= E_{pk}\left(\frac{M_{0,1} * M_{1,2}}{M_{0,1}}\right) \\
&= E_{pk}\left(\frac{M_3}{M_2}\right) \\
&= E_{pk}(M_2')
\end{aligned}$$

Scenario 2: On the other hand, when $h > 3$, A and T jointly involve in the SMP protocol (Step 14 in Algorithm 1). Following from the SMP protocol (as given in Algorithm 2), the value of Φ_l, for $1 \leq l \leq h-2$, can be formulated as shown below:

$$\begin{aligned}
\Phi_l &= E_{pk}\left(\prod_{j=1}^{h-l-1} x_j\right) \\
&= E_{pk}\left(\prod_{j=1}^{h-l-1} M_{h-j-1,h-j}\right) \\
&= E_{pk}\left(\prod_{k=l}^{h-2} M_{k,k+1}\right) \\
&= E_{pk}\left(\frac{M_{0,1} * \ldots * M_{h-2,h-1}}{M_{0,1} * \ldots * M_{l-1,l}}\right) \\
&= E_{pk}\left(\frac{M_h}{M_{l+1}}\right) \\
&= E_{pk}(M_{l+1}')
\end{aligned}$$

□

3.5.2. Phase 2 - Secure Computation of Recommendation Scores.

During Phase 2, A with input Φ along with T and the internal users jointly compute

the recommendation score for each potential candidate. The main steps involved in Phase 2 of the PFR protocol are shown as steps 17 to 37 in Algorithm 1. To start with, initially A computes a vector α (which is different for each B_i) of size h as follows:

$$\begin{aligned} \alpha_1 &= E_{pk}(C_{A,B_i}) \\ \alpha_l &= \Phi_{l-1}^{C_{A,B_i}} \bmod N^2, \text{for } 2 \leq l \leq h \end{aligned}$$

After this, A sends A, Φ, and corresponding α to B_i, for $1 \leq i \leq m$. Then, each internal user U_j receives the values of A, Φ, and α from $Pr(U_j)$ and checks whether A is already a friend of U_j (this case happens only if U_j is equal to one of the B_i's). If $A \in F(U_j)$, then U_j simply forwards A, Φ, and α to each of his/her child friend. Otherwise, U_j computes the encryption of shares of his/her recommendation score as below:

$$\beta_j = \alpha_1^{D_{U_j}} \bmod N^2; \;\; \gamma_j = \alpha_2 * \Phi_1^{C_{Y,U_j}} \bmod N^2$$

where D_{U_j} denotes the degree of U_j (i.e., $|F(U_j)|$) and Y is the parent friend of U_j. After this, U_j sends $Z_j = \{E_{pk}(U_j), \langle \beta_j, \gamma_j \rangle\}$ to A. Note that U_j can receive multiple pairs of (Φ, α) which occurs only when there exist multiple shortest paths from A to U_j. Under this scenario, U_j creates the encrypted partial scores for each pair of (Φ, α) and simply appends them to Z_j as follows.

$$Z_j = \{E_{pk}(U_j), \langle \beta_{1,j}, \gamma_{1,j} \rangle, \ldots, \beta_{s,j}, \gamma_{s,j} \rangle\}$$

where each $\beta_{l,j}, \gamma_{l,j}$, for $1 \leq l \leq s$, is computed as explained above for each pair of (Φ, α) and s denotes the number of such pairs (number of shortest paths to U_j from A). In addition, if the counter (t) corresponding to U_j is greater than 0, then U_j generates necessary information for his/her child friends as follows.

- Update Φ and α:

- $\Phi_l = \Phi_{l+1}$, for $1 \leq l \leq t$
- $\alpha_1 = \alpha_1^{C_{Y,U_j}} \bmod N^2$
- $\alpha_l = \alpha_{l+1} * \Phi_{l-1}^{C_{Y,U_j}}$, for $2 \leq l \leq t+1$

- Send A, Φ and α to his/her child friends. If U_j receives multiple pairs of (Φ, α), U_j updates each pair as above and sends all updated pairs to the child friends.

Upon receiving the entries from all potential candidates, A and T involve in a secure multiplication and addition (SMPA) protocol. The main steps involved in the SMPA protocol are shown in Algorithm 3. Without loss of generality, consider the entry $Z_j = \{E_{pk}(U_j), \langle \beta_{1,j}, \gamma_{1,j} \rangle, \ldots, \langle \beta_{s,j}, \gamma_{s,j} \rangle\}$, where s denotes the number of shortest paths from A to U_j. In addition, let $\beta_{k,j} = E_{pk}(a_{k,j})$ and $\gamma_{k,j} = E_{pk}(b_{k,j})$, for $1 \leq k \leq s$. The goal of the SMPA protocol is to securely compute $a_{1,j} * b_{1,j} + \cdots + a_{s,j} * b_{s,j}$ as output without revealing the values of $a_{k,j}$ and $b_{k,j}$, for $1 \leq k \leq s$, to either A or T. At the end of the SMPA protocol, only user A knows the recommendation score corresponding to U_j, for $1 \leq j \leq n$. The basic idea of the SMPA protocol is based on the following property which holds for any given $a_{k,j}, b_{k,j} \in \mathbb{Z}_N$, for $1 \leq k \leq s$:

$$\sum_{k=1}^{s} a_{k,j} * b_{k,j} = \sum_{k=1}^{s} (a_{k,j} + r_{k,j}) * (b_{k,j} + r'_{k,j}) - \sum_{k=1}^{s} a_{k,j} * r'_{k,j} - \sum_{k=1}^{s} b_{k,j} * r_{k,j} - \sum_{k=1}^{s} r_{k,j} * r'_{k,j}$$

where $r_{k,j}$ and $r'_{k,j}$ are random numbers in $\mathbb{Z}_N$ and all arithmetic operations are performed under modulo N. The overall steps involved in SMPA are shown in Algorithm 3. Initially, A randomizes each encrypted tuple $\langle \beta_{k,j}, \gamma_{k,j} \rangle$, for $1 \leq k \leq s$, as follows:

$$\begin{aligned} \widetilde{\beta}_{k,j} &= \beta_{k,j} * E_{pk}(r_{k,j}) \bmod N^2 \\ \widetilde{\gamma}_{k,j} &= \gamma_{k,j} * E_{pk}(r'_{k,j}) \bmod N^2 \end{aligned}$$

Here $r_{k,j}$ and $r'_{k,j}$ are randomly chosen in $\mathbb{Z}_N$. A also randomizes $E_{pk}(U_j)$ and performs these homomorphic operations (steps 1(b) to 1(g) of Algorithm 3). The r_j and $\tilde{r}_j$ are

Algorithm 3 SMPA

Require: A's input is Z_j

1: A:

(a). **for** $1 \leq k \leq s$ **do**:

- $\widetilde{\beta}_{k,j} \leftarrow \beta_{k,j} * E_{pk}(r_{k,j}) \bmod N^2$, where $r_{k,j} \in \mathbb{Z}_N$
- $\widetilde{\gamma}_{k,j} \leftarrow \gamma_{k,j} * E_{pk}(r'_{k,j}) \bmod N^2$, where $r'_{k,j} \in \mathbb{Z}_N$

(b). $\lambda_j \leftarrow E_{pk}(U_j) * E_{pk}(r_j) \bmod N^2$, where $r_j \in \mathbb{Z}_N$

(c). $E_{pk}(r) \leftarrow E_{pk}(\sum_{k=1}^{s} r_{k,j} * r'_{k,j})$

(d). $E_{pk}(r_1) \leftarrow \prod_{k=1}^{s} \beta_{k,j}^{r'_{k,j}} \bmod N^2$

(e). $E_{pk}(r_2) \leftarrow \prod_{k=1}^{s} \gamma_{k,j}^{r_{k,j}} \bmod N^2$

(f). $\tau \leftarrow E_{pk}(\tilde{r}_j) * E_{pk}(r)^{N-1} \bmod N^2$, where $\tilde{r}_j \in \mathbb{Z}_N$

(g). $w_j = \tau * E_{pk}(r_1)^{N-1} * E_{pk}(r_2)^{N-1} \bmod N^2$

(h). Send w_j, λ_j and $\widetilde{\beta}_{k,j}, \widetilde{\gamma}_{k,j}$, for $1 \leq k \leq s$ to T

2: T:

(a). Receive parameters from A

(b). $\widetilde{a}_{k,j} \leftarrow D_{pr}(\widetilde{\beta}_{k,j})$; $\widetilde{b}_{k,j} \leftarrow D_{pr}(\widetilde{\gamma}_{k,j})$, for $1 \leq k \leq s$

(c). $c_j \leftarrow \sum_{k=1}^{s} \widetilde{a}_{k,j} * \widetilde{b}_{k,j} \bmod N$

(d). $z_j \leftarrow D_{pr}(w_j)$; $s_{1,j} \leftarrow z_j + c_j \bmod N$

(e). $s_{2,j} \leftarrow D_{pr}(\lambda_j)$; send $s_{1,j}$ and $s_{2,j}$ to A

3: A:

(a). Receive $s_{1,j}$ and $s_{2,j}$ from T

(b). $\widetilde{RS}(A, U_j) \leftarrow s_{1,j} - \tilde{r}_j \bmod N$ (recommendation score)

(c). $U_j \leftarrow s_{2,j} - r_j \bmod N$ (corresponding user ID)

---

also random numbers in $\mathbb{Z}_N$. Then, A sends $\widetilde{\beta}_{k,j}$ and $\widetilde{\gamma}_{k,j}$, for $1 \leq k \leq s$, to T along with w_j and λ_j. Upon receiving, T decrypts $\widetilde{\beta}_{k,j}$ and $\widetilde{\gamma}_{k,j}$, for $1 \leq k \leq s$, multiplies and adds them as below:

- For $1 \leq k \leq s$, $\widetilde{a}_{k,j} = D_{pr}(\widetilde{\beta}_{k,j})$ and $\widetilde{b}_{k,j} = D_{pr}(\widetilde{\gamma}_{k,j})$

- $c_j = \sum_{k=1}^{s} \widetilde{a}_{k,j} * \widetilde{b}_{k,j} \bmod N$.

Furthermore, T decrypts w_j and λ_j: $z_j = D_{pr}(w_j)$ and $s_{2,j} = D_{pr}(\lambda_j)$, and computes $s_{1,j} = z_j + c_j \bmod N$. Then, T sends $s_{1,j}$ and $s_{2,j}$ to A. Finally, A removes the randomness from $s_{1,j}$ and $s_{2,j}$ to get the actual score and user ID U_j as follows:

$$\widetilde{RS}(A, U_j) = s_{1,j} - \tilde{r}_j \bmod N; \;\; U_j = s_{2,j} - r_j \bmod N$$

Here, $\widetilde{RS}(A, U_j)$ is the recommendation score for user U_j based on Equation 3.5. Note that $(N-1)$ represents "-1" under $\mathbb{Z}_N$.

Theorem 2. *The output of Phase 2 is the list of recommendation scores along with the corresponding users IDs. That is, for any given entry Z_j:*

$$\begin{aligned} s_{1,j} - \tilde{r}_j \bmod N &= \widetilde{RS}(A, U_j) \\ s_{2,j} - r_j \bmod N &= U_j \end{aligned}$$

Where $s_{1,j}$ and $s_{2,j}$ are the final values sent to A from T corresponding to the entry Z_j in the SMPA protocol, for $1 \leq j \leq n$.

Proof. Without loss of generality, consider a potential user U_j who receives A and (Φ, α) pairs from his/her parent friends. Let us assume that U_j receives s number of different (Φ, α) pairs (representing s number of shortest paths from A to U_j) and let $\beta_{k,j}, \gamma_{k,j}$ denote the encrypted partial scores corresponding to k^{th} pair (Φ_k, α_k) (denoting k^{th} shortest path from A to U_j), for $1 \leq k \leq s$. U_j computes the encrypted partial shares for k^{th} pair as follows:

$$\begin{aligned} \beta_{k,j} &= \alpha_{1,k}^{D_{U_j}} \bmod N^2 = E_{pk}\left(D_{U_j} * \prod_i C_{S_{i-1},S_i}\right) \\ \gamma_{k,j} &= \alpha_{2,k} * \Phi_{1,k}^{C_{Y,U_j}} \bmod N^2 = E_{pk}(M'_l * |P_k(A, U_j)|) \end{aligned}$$

where $\alpha_{y,k}$ (resp., $\Phi_{y,k}$) denotes the y^{th} component of vector α_k (resp., Φ_k); $i = 1, \ldots, l-1$; $l = L(U_j)$ and $S_{i-1} = Pr(S_i)$ along the k^{th} path from A to U_j. Then, U_j sends $Z_j = \{E_{pk}(U_j), \langle \beta_{1,j}, \gamma_{1,j} \rangle, \ldots, \langle \beta_{s,j}, \gamma_{s,j} \rangle\}$ to A. Upon receiving, A and T involve in the SMPA protocol. As mentioned earlier, let $\beta_{k,j} = E_{pk}(a_{k,j})$ and $\gamma_{k,j} = E_{pk}(b_{k,j})$, for $1 \leq k \leq s$. Since SMPA securely multiplies each $(\beta_{k,j}, \gamma_{k,j})$ pair and then adds them, the output of the SMPA protocol can be formulated as follows:

$$\begin{aligned} s_{1,j} - \tilde{r}_j \bmod N &= \sum_{k=1}^{s} a_{k,j} * b_{k,j} \\ &= \sum_{k=1}^{s} (D_{U_j} * \prod_{i} C_{S_{i-1},S_i}) * (M'_l * |P_k(A, U_j)|) \\ &= M'_l * \sum_{k=1}^{s} \left(|P_k(A, U_j)| * \prod_{i} C_{S_{i-1},S_i} \right) * D_{U_j} \\ &= \widetilde{RS}(A, U_j) \end{aligned}$$

Similarly, it is easy to show that $s_{2,j} - r_j \bmod N = U_j$. □

During the actual implementation, the SMPA protocol can be initiated in parallel as the computation for potential user U_j is independent of others. Thus, overall, SMPA requires only one round of communication between A and T.

Example 2. *As an example, various intermediate steps and results involved in the PFR protocol based on Figure 3.2 are shown. Here $h = 3$ and Lee is the target user. Following from initialization step, users at 1-hop away from Lee, that is, $\langle Hall, Cox, Bell \rangle$ have a value of $t = 2$. Similarly, $\langle Ford, Butler, Cole, Kelly \rangle$ have a value of $t = 1$. Whereas, $\langle Shaw, Ray, Fox, Ryan, Hart, Jones \rangle$ have $t = 0$. Each of them is aware of pk and also their parent and child friends (following from the initialization step).*

Phase 1: *Initially, Hall computes $L_{Hall}[1] = E_{pk}(C_{Hall,Ford} + C_{Hall,Butler}) = E_{pk}(3)$. Similarly, Cox and Bell compute $L_{Cox}[1] = E_{pk}(7)$ and $L_{Bell}[1] = E_{pk}(3)$*

respectively. Observe that $C_{Hall,Cox}$ *is not included in* $L_{Hall}[1]$ *and* $L_{Cox}[1]$ *since Hall and Cox are at same level from Lee. After this, Hall, Cox, and Bell send* L_{Hall}, L_{Cox}, *and* L_{Bell} *resp., to Lee. Upon receiving values, Lee computes* $L_{Lee}[1] = L_{Hall}[1] * L_{Cox}[1] * L_{Bell}[1] \bmod N^2 = E_{pk}(13)$. *Then, Lee sets the encrypted scalar factors as follows:*

$$\Phi_1 = L_{Lee}[1] = E_{pk}(13); \;\; \Phi_2 = E_{pk}(1)$$

Phase 2: *During Phase 2, Lee computes encrypted vector* α *(different) for each of his friends. Without loss of generality, consider user Hall. Lee creates* α *for Hall as follows.*

$$\begin{aligned} \alpha &= \langle E_{pk}(C_{Lee,Hall}), \Phi_1^{C_{Lee,Hall}}, \Phi_2^{C_{Lee,Hall}} \rangle \\ &= \langle E_{pk}(2), E_{pk}(2 * 13), E_{pk}(2 * 1) \rangle \end{aligned}$$

Then, Lee sends $\langle Lee, \Phi, \alpha \rangle$ *to Hall who further forwards them to Ford and Butler. The final entries (that are sent to Lee) from all potential users are shown in Table 3.2. Finally, Lee and T involve in the SMPA protocol to get the scaled recommendation scores. E.g., the recommendation score for Ford is* $\widetilde{RS}(Lee, Ford) = 2 * D_{Ford} * 4 * 13 = 104 * D_{Ford}$. *It is clear that* $\widetilde{RS}(Lee, Ford) = M_h * TN * RS$, *where actual recommendation score for Ford is* $RS = 4 * \frac{2}{5} * \frac{D_{Ford}}{TN}$ *and* $M_h = 65$. □

3.5.3. Security Analysis. This sub-section analyzes the security of the initialization step and each phase in the proposed PFR protocol separately.

First, during the initialization step, the generation of candidate network does not leak any information. Consider the two possible cases of the initialization step as discussed in Observation 2. For case 1, where U receives the same counter values from multiple parents (on the same level), U cannot predict whether there exists a friendship between any two parents. For example, suppose U receives $t = 2$ from two

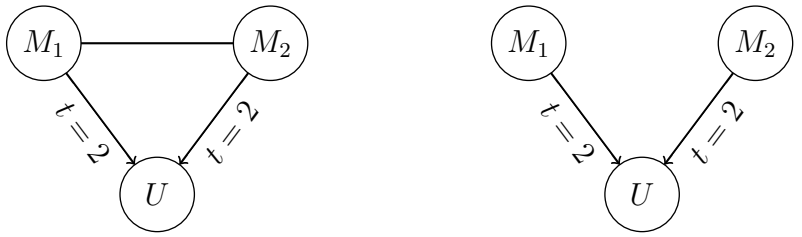
Table 3.2: Encrypted partial scores corresponding to each potential candidate based on the PFR protocol

$\{E_{pk}(Ford), \langle E_{pk}(2 * D_{Ford}), E_{pk}(52)\rangle\}$	$\{E_{pk}(Butler), \langle E_{pk}(2 * D_{Butler}), E_{pk}(39)\rangle,$ $\langle E_{pk}(2 * D_{Butler}), E_{pk}(78)\rangle\}$
$\{E_{pk}(Cole), \langle E_{pk}(2 * D_{Cole}), E_{pk}(65)\rangle\}$	$\{E_{pk}(Fox), \langle E_{pk}(2 * D_{Fox}), E_{pk}(7)\rangle,$ $\langle E_{pk}(6 * D_{Fox}), E_{pk}(10)\rangle\}$
$\{E_{pk}(Kelly), \langle E_{pk}(D_{Kelly}), E_{pk}(52)\rangle\}$	$\{E_{pk}(Ryan), \langle E_{pk}(2 * D_{Ryan}), E_{pk}(5)\rangle,$ $\langle E_{pk}(6 * D_{Ryan}), E_{pk}(8)\rangle\}$
$\{E_{pk}(Shaw), \langle E_{pk}(4 * D_{Shaw}), E_{pk}(6)\rangle\}$	$\{E_{pk}(Hart), \langle E_{pk}(6 * D_{Hart}), E_{pk}(10)\rangle,$ $\langle E_{pk}(3 * D_{Hart}), E_{pk}(6)\rangle\}$
$\{E_{pk}(Ray), \langle E_{pk}(4 * D_{Ray}), E_{pk}(5)\rangle\}$	$\{E_{pk}(Jones), \langle E_{pk}(3 * D_{Jones}), E_{pk}(8)\rangle\}$

parents M_1 and M_2. Here U cannot distinguish the two scenarios shown in Figure 3.3. This is because the value of t received by U is independent of the relationship between the parents who sent it.

Similarly, for case 2, where U can receive different counter values from multiple parents (different levels), U cannot deduce any information by simply using the received counter values. For example, consider that U receives $t = 3$ from M_1 and $t = 2$ from M_2. Then, U cannot distinguish between the following two possible scenarios as shown in Figure 3.4, where J is the parent user of M_2. It is clear that the counter values passed to U are independent of the relationship between M_1 and M_2. Hence, passing the counter values during the formation of candidate network (initialization step) in PFR does not reveal any information.

During Phase 1, each internal user sends the encrypted aggregated data only to $Pr(U_j)$. Thus, the privacy of individual users is preserved as per the security definition of SMC [50]. In addition, during the SMP protocol, A first randomizes the values of L_A and sends them to T. Therefore, the simulated view of T is indistinguishable compared to the real view (trusted third party model). Furthermore, since T sends only the encrypted scalar factors to A, neither the values of $M_{i-1,i}$'s nor M_i''s are revealed to A, for $1 \leq i \leq h-1$. Therefore, the privacy of A and U_j are preserved, for $1 \leq j \leq n$. Note that the scalar factor for users at level h is always



(a) M_1 and M_2 are friends

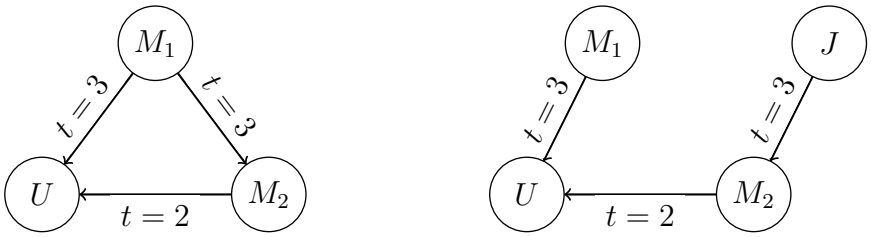
(b) M_1 and M_2 are not friends

Figure 3.3: Case 1 of initialization step in PFR

known to A, since $M'_h = 1$ always hold. However, it is worth pointing out that this does not reveal any information to A.

On the other hand, in Phase 2, A initially sends $\{A, \Phi, \alpha\}$ to each B_i, for $1 \leq i \leq m$. Each internal user U_j, computes his/her encrypted partial scores using D_{U_j} and C_{Y_i,U_j}, where Y_i is the parent friend of U_j (with multiple parents denoting multiple shortest paths from A to U_j). Then, U_j sends his entry Z_j in encrypted form to A. Here, the privacy of each U_j is preserved under the assumption that number of shortest paths to U_j can be revealed to A. However, this problem can be solved by random masking without affecting the recommendation score (more details are given in the later part of this section). During the SMPA protocol, the values of each entry are randomized in $\mathbb{Z}_N$ and sent to T. That is, the values of $D_{U_j} * \prod_i C_{S_{i-1},S_i}$ and $M'_l * |P_k(A, U_j)|$, for $1 \leq k \leq s$, are randomized and sent to T. Therefore, the privacy of A and U_j is preserved. In addition, the final output sent to A is the actual output and the intermediate values are never revealed to A, T and other internal users.

Based on the above discussions, it is clear that, apart from the initialization step, Phase 1 and 2 are secure. In addition, the values returned from Phase 1 to 2 are pseudo-random; therefore, the sequential composition of the two Phases lead to a secure protocol according to the Composition Theorem [49].



(a) M_1 and M_2 are friends (b) M_1 and M_2 are not friends

Figure 3.4: Case 2 of initialization step in PFR

3.5.4. Complexity Analysis. The computation and communication costs of each party in the PFR protocol are analyzed. For the rest of this sub-section, this work omits the cost of the Initialization step since they are negligible compared to the encryption costs in Phase 1 and 2 of PFR.

Computation Cost. For Phase 1, the computation cost of each internal user U_j depends on his/her counter value and number of child friends. In addition, irrespective of the counter value, U_j has to perform one encryption operation. Therefore, the computation complexity of U_j is bounded by one encryption and $O(t * |Ch(U_j)|)$ homomorphic addition operations. Whereas, the computation complexity of A mainly depends on h and m. If $h = 2$, then A simply performs one encryption operation. However, when $h = 3$, A's computation complexity is bounded by $O(h * m)$ homomorphic additions and one encryption. On the other hand, if $h > 3$, the computation complexity of A mainly comes from the SMP protocol which depends on the value of h. That is, A's computation complexity is bounded by $O(h * m)$ homomorphic additions and $O(h)$ number of encryption operations. Whereas, the computation complexity of T is bounded by $O(h)$ decryption operations (coming from the SMP protocol).

In Phase 2, the computation complexity of each internal user (excluding B_i's) depends on his/her t and s (number of shortest paths from A to U_j). Specifically,

U_j's computation cost is bounded by $O(t * s)$ exponentiations and homomorphic additions. On the other hand, A has to initially compute α, which depends on the value of h, for each B_i. Therefore, the computation complexity of A for computing all α values is bounded by $O(h * m)$ encryption and exponentiation operations. In addition, during the SMPA protocol, A has to randomize all components of each potential candidate. Let n denote the number of potential candidates and s be the maximum number of shortest paths, then the computation cost of A in the SMPA protocol is bounded by $O(s * n)$ encryption and exponentiation operations. Overall, during Phase 2, the computation complexity of A is bounded by $O(s * n)$ encryption and exponentiation operations (under the assumption $s * n > h * m$). Whereas, the computation complexity of T is bounded by $O(s * n)$ decryption operations (coming from SMPA).

Communication Cost. Without loss of generality, let K denote the Paillier encryption key size (in practice, K should be at least 1,024 bits). During Phase 1, the communication complexity between any two internal users is bounded by $O(K*t)$ bits. Note that t may vary between each pair of users depending on their location in the corresponding candidate network and only adjacent users (i.e., friends) communicate to each other. Whereas, between A and all B_i's, the communication cost is bounded by $O(K*h*m)$ bits. In addition, for the SMP protocol, the communication complexity between A and T is bounded by $O(K * h)$ bits.

Additionally, during Phase 2, the communication cost between any two internal users is bounded by $O(K * t)$ bits, where t is the counter of the corresponding parent user. Since A has to send Φ and α to each B_i, for $1 \leq i \leq m$, the communication cost between A and all B_i's is bounded by $O(K * h * m)$ bits. In addition, since each potential candidate sends the encrypted partial shares to A, the communication cost between A and all potential candidates is bounded by $O(K*s*n)$ bits (assuming there exist s number of shortest paths from A to each potential candidate). Finally,

during the SMPA protocol, the communication cost between A and T is bounded by $O(K * s * n)$ bits.

3.6. PRACTICAL IMPLEMENTATION DETAILS

3.6.1. Masking Number of Shortest Paths. As mentioned earlier, the PFR protocol reveals (only to A) the number of shortest paths from A to each potential candidate U_j, for $1 \leq j \leq n$. However, it is unclear how this additional information affects the privacy of U_j. Nevertheless, this problem can be solved by randomly masking the number of entries corresponding to each U_j without affecting his/her final recommendation score. Suppose $Z_j = \{E_{pk}(U_j), \langle \beta_{1,j}, \gamma_{1,j} \rangle, \ldots, \langle \beta_{s,j}, \gamma_{s,j} \rangle\}$ is the entry corresponding to U_j, where s denotes the number of shortest paths from A to U_j in the corresponding candidate network. Briefly, the steps involved in masking the entry Z_j by U_j are as given below, for $1 \leq j \leq n$:

- Randomly mask the number of shortest paths by computing Z'_j as follows.

$$Z'_j = \{E_{pk}(U_j), \langle \beta_{1,j}, \gamma_{1,j} \rangle, \ldots, \beta_{s+\theta,j}, \gamma_{s+\theta,j} \rangle\}$$

  where θ is the security parameter chosen by U_j, such that $\beta_{s+i,j} = \gamma_{s+i,j} = E_{pk}(0)$, for $1 \leq i \leq \theta$. Note that the encryption scheme used in this work (based on HPEnc$^+$ system) is probabilistic. Therefore, encryption of 0 each time yields different (random) ciphertext in $\mathbb{Z}_{N^2}$.

- Send the masked entry Z'_j to A

The rest of the steps are same as in Phase 2 of PFR. Observe that applying SMPA on Z'_j yields the same result as on Z_j, for $1 \leq j \leq n$.

3.6.2. Data Encryption and Secure Peer-to-Peer Communication. The PFR protocol assumes there exist peer-to-peer network connectivity and the

communication between any two users is secure. However, in practice, as it is the case is many online social networks, communication between any two users should happen through the social network provider (say the network administrator) for security reasons. Another reason for this is a user may not be online at the receiving end.

In this work, users profile data are assumed to be encrypted first (using his/her own secret key) and then stored on the server of network administrator [57]. From user's perspective, this assumption, which is also practical, gives more flexibility to them since it gives more control to users on their own data. In addition, secure communication between any two users can be achieved by establishing a secure session (using AES session key [58]) between them. During the process of establishing a secure session if the end user is offline, then the information corresponding to the session key (in encrypted form using public key of end user) is stored on the network administrator's server. Under this scenario, the network administrator merely acts as an intermediate router who simply stores the encrypted data sent by the sender and delivers them to the concerned end-user after he/she logins. Note that the data to be sent is first encrypted using the corresponding session key of sender and then stored on the server. Therefore, only the end-user who holds the session key can decrypt it. For example, during the initialization step of PFR, each user U first establishes a secure session with his/her friends. Then, the value of t to be sent is encrypted and stored on the server. Once the intended end-users (i.e., friends of U) logins into social network, the network administrator sends the encrypted value of t to him/her who decrypts it to know his/her counter value. Note that the session keys should be changed occasionally for security reasons.

3.7. EXTENSION TO PFR

In many online social networks, such as Facebook, there exist an inherent security issue due to the information flow between different entities. For example, consider the scenario of user U sending a message to another user V. Though the message is encrypted using the session key of U, as explained above, it is clear that the network administrator will know that U and V have some kind of relationship. This is because of the fact that the communication between any two users should pass through the network administrator. Here U and V might be friends but this may not be the case always since a user can send a message to any other user in the social network (i.e., no need of friendship between the two users).

In particular to the PFR protocol, this information might be too specific since U sends an encrypted counter or some other information during Phase 1 or 2 to only his/her parent or child nodes (i.e., friends of U). However, the network administrator cannot distinguish which message belongs to which application. This is because the messages are encrypted and thus the underlying application is not transparent to the network administrator. Nevertheless, the network administrator will still know that there exists some kind of relationship between U and the users at the receiving end. It is unclear how this extra information will be useful to the network administrator in deducing any private information of U.

As mentioned above, the PFR protocol might leak the additional information that there exists some kind of relationship between U and his/her friends to the network administrator. Note that this kind of information is never revealed to other users in the PFR protocol. If the message flow is always between U and only his/her friends, then the probability of guessing user $V_j \in F(U)$ is a friend of U by the network administrator is very high. Therefore, to mitigate this issue or to provide better security, this sub-section presents an extended version (denoted by PFR_{rand}) of the

PFR protocol by randomly including additional users (apart from the actual friends) to take participation in the PFR protocol without affecting the final recommendation scores. By doing so, this work actually allows users to randomize their friend lists; therefore, reducing the probability of guessing the friends of a user by the network administrator.

Similar to PFR, the PFR_{rand} protocol consists of an initialization step and two phases. However, the methodology is slightly different from PFR. Therefore, the key steps in PFR_{rand} that are different from PFR are discussed below.

3.7.1. Initialization Step. To start with, the target user A selects a random set of dummy friends (excluding the actual friends of A) from the social network, denoted by $D(A)$, where $|D(A)|$ is the security parameter for A. Note that the dummy friends can also be referred to as dummy child nodes. Then, A sets the counter t to $h-1$ and sends (δ, t) to each user Y_j in $F(A) \cup D(A)$. Where $\delta = 1$ if $Y_j \in F(A)$, and 0 otherwise. Then, each intermediate user U selects his/her random set of dummy users $D(U)$, stores (δ, t) and the parent user who sent the entry to U locally (as explained below). In addition, he/she sends the updated (δ, t) entry to users in $Ch(U) \cup D(U)$, where $Ch(U)$ denotes the actual child nodes of U. This process is continued until the users at h-hop away from A receive a counter of $t = 0$. Since U can receive multiple pairs of (δ, t), depending on whether U is a dummy user or not, this work addresses the following two cases.

Case 1: If the δ values received by U are all 0's (whereas the values of t can be different), then U is actually a dummy user (i.e., not part of candidate network). Under this case, U stores δ as 0 and the maximum t value received locally. Plus, U selects the user who sent the maximum t as his/her parent (i.e., $P(U)$). In addition, U sends (δ, t) to each user in $Ch(U) \cup D(U)$ with $\delta \leftarrow 0$ and $t \leftarrow t-1$.

Case 2: On the other hand, if U receives either different δ values (i.e., both 0 and 1's) or same δ values of 1, then U is part of the candidate network. Therefore,

U stores δ as 1 and selects the maximum t value among the entries with $\delta = 1$ as his/her counter value . Also, among the users who sent $\delta = 1$, U selects the user who sent the maximum t as $Pr(U)$. Unlike in the previous case, U sends $(1, t)$ to users in $Ch(U)$ and $(0, t)$ to users in $D(U)$, where $t \leftarrow t - 1$.

At the end of the initialization step, each user who participates in the PFR_{rand} protocol knows whether he/she is a dummy user (i.e., $\delta = 0$) or part of the candidate network ($\delta = 1$). In addition, each user knows his/her parent user ($Pr(U)$), actual child users ($Ch(U)$), and dummy child users ($D(U)$).

3.7.2. Phase 1 - Secure Computation of Scalar Factors. Following from Case 1 and 2 of the above initialization step, one can deduce the following observation in the PFR_{rand} protocol.

Observation 3. *For any internal user U who belongs to the candidate network (i.e., $\delta = 1$), $Pr(U)$ belongs to the candidate network and is also the same as in the PFR protocol (assuming single parent).*

The basic idea of Phase 1 in PFR_{rand} is same as in PFR. However, the only difference is that whenever an internal node (and also A) receives the encrypted aggregated data from his/her child nodes (i.e., users in $Ch(U) \cup D(U)$), U simply dumps the messages received from dummy child nodes ($\in D(U)$). More specifically, only the messages received from actual child nodes are used in performing further computation by U and the resulting aggregated data is forwarded to $Pr(U)$. Since the messages from dummy nodes are not used for computations and following from Observation 3, it is clear that the final output from Phase 1 is same as that of Phase 1 in PFR.

3.7.3. Phase 2 - Secure Computation of Recommendation Scores. Similar to PFR, at the end of Phase 1 in PFR_{rand}, A has the list of encrypted scalar factors (i.e., Φ) for all potential users within a radius of h. Note that, as mentioned above, these scalar factors are computed based on the candidate network. That is,

even though dummy child friends are added for each user to provide better security, the corresponding messages are not used during the computation of encrypted scalar factors for each level. Thus, the final result of Phase 1 in PFR_{rand} is equivalent to Φ (list of encrypted scalar factors for each level).

During Phase 2, A initially sends his/her ID, Φ, and α (which is computed similar to in PFR) to each user in $F(A) \cup D(A)$. Then, each participating user M_j in PFR_{rand}, after receiving encrypted data from his/her parent(s), computes the encrypted shares of his/her recommendation score as mentioned in PFR. After this, M_j sends his/her entry Z_j, as defined below, to A.

$$Z_j = \{E_{pk}(M_j), E_{pk}(\delta_j), \langle \beta_{1,j}, \gamma_{1,j} \rangle, \ldots, \beta_{s,j}, \gamma_{s,j} \rangle\}$$

where s denotes the number of shortest paths from A to M_j and the flag δ_j denotes whether or not M_j is part of the candidate network. Observe that δ_j is not used in the PFR protocol. Note that, similar to PFR, the number of shortest paths can be masked by M_j before sending it to A. For each entry Z_j received, A first retrieves δ_j securely as follows:

- A randomizes $E_{pk}(\delta_j)$ by computing $\sigma_j = E_{pk}(\delta_j) * E_{pk}(r_j) \bmod N^2$, where r_j is a random number chosen from $\mathbb{Z}_N$, and sends σ_j to T.
- Upon receiving, T computes $\delta'_j = D_{pr}(\sigma_j)$ and sends δ'_j to A.
- A removes the randomization from δ'_j to get δ_j, i.e., computes $\delta_j = \delta'_j - r_j \bmod N$.

At the end of this step, A knows which entries belong to actual potential users (i.e., $\delta_j = 1$) and dummy users (i.e., $\delta_j = 0$). Finally, A and T involve in the SMPA protocol to get the recommendation scores and corresponding user IDs for only those

entries with $\delta_j = 1$. Note that, the number of instantiations of SMPA in PFR_{rand} is the same as in PFR.

3.8. EMPIRICAL ANALYSIS

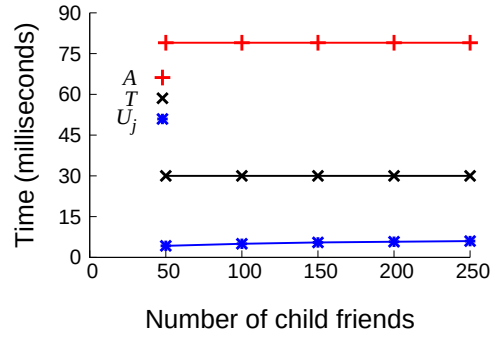
Since the effectiveness of PFR is the same as CSM method [14], this subsection analyzes the computation costs of PFR based on different parameters. In addition, the computation costs of PFR_{rand} are compared with that of PFR.

3.8.1. Platform and Dataset Description. The experiments used Paillier cryptosystem [55] as the underlying encryption scheme. The proposed protocols were implemented in C, and experiments were conducted on an Intel® Xeon® Six-Core™3.07GHz PC with 12GB memory running Ubuntu 10.04 LTS.

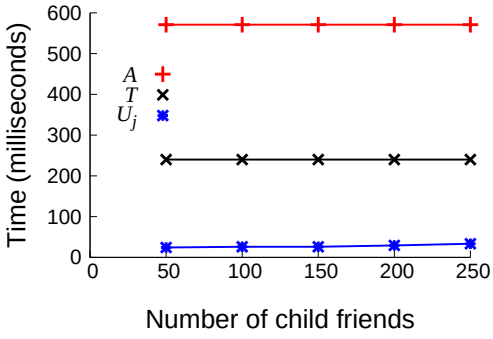
Since it is hard to control parameters in a real-world dataset, this work simulated the environment and computed the computation costs. In all the experiments, the minimum number of messages exchanged between any two users U and V (i.e., $C_{U,V}$) is assumed to be uniformly distributed in [1, 1000]. Also, this work assumes that there is no communication delay between participating users (which further implies that users are online). In addition, the number of child friends for each user (including the target user A) is varied from 50 to 250.

3.8.2. Performance of PFR. First, the computation costs of A, T, and internal user U_j in Phases 1 and 2 of PFR are analyzed separately. Since the run time of U_j depends on which level he/she belongs to, the average over the computation costs of all internal users at different levels are presented. For the rest of this subsection, the Paillier's encryption key size (i.e., K) is fixed to either 1024 or 2048 bits. The results are as shown in Figure 3.5.

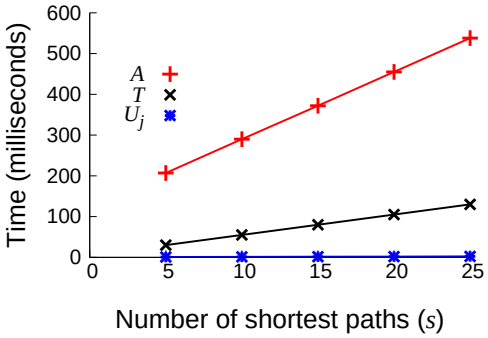
For Phase 1, the value of h is set to 6 and the run time of A, T, and U_j are computed by varying the number of child friends from 50 to 250 (note that users



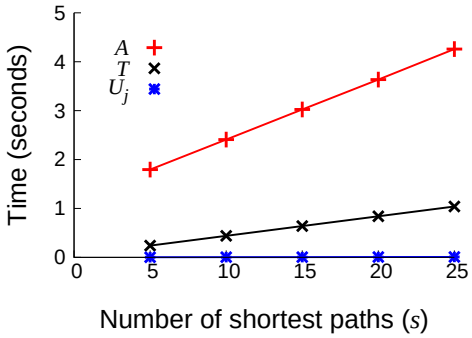
(a) Cost of Phase 1 for $K = 1024$



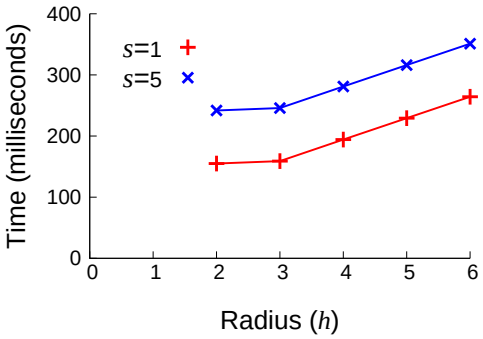
(b) Cost of Phase 1 for $K = 2048$



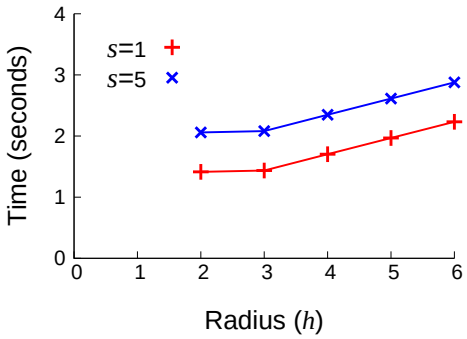
(c) Cost of Phase 2 for $K = 1024$



(d) Cost of Phase 2 for $K = 2048$



(e) Total cost of PFR for $K = 1024$



(f) Total cost of PFR for $K = 2048$

Figure 3.5: Computation costs of PFR

at level $h-1$ and h do not involve in Phase 1). As shown in Figure 3.5(a), for Paillier key size K=1024 bits, the computation time for A, T, and U_j are 79, 30, and 4.25 milliseconds respectively when the number of child friends is 50. In addition, the computation time of U_j varies only slightly (due to less expensive homomorphic

operations) from 4.25 to 6 milliseconds when the number of child friends of U_j are varied from 50 to 250. However, for A the computation time remains the same since the cost of homomorphic addition operations are negligible compared to the encryption operations involved in SMP. Since h is fixed, the encryption cost in SMP remains the same irrespective of the child friends of A. Therefore, the computation costs of A and T remains the same in Phase 1. A similar trend can be observed for K=2048 bits as shown in Figure 3.5(b). Briefly, when the number of child friends is 50, the computation costs of A, T, and U_j are 571, 240, and 24.25 milliseconds respectively. Also, irrespective of the number of child friends and h, the computation time of U_j is always significantly less than that of A and T.

During Phase 2, the computation time to find the recommendation score for each potential candidate U_j mainly depends on m (number of A's friends) and the cost of SMPA which in turn depends on the number of shortest paths (s) from A to U_j. However, for any given m, the cost to compute recommendation score for each U_j varies with the corresponding s. Thus, the computation costs for A, T and U_j based on varying values of s, with $h = 6$ and $m = 50$, are analyzed. As shown in Figure 3.5(c), the computation cost of U_j (averaged over different levels) increases from 0.8 to 2.4 milliseconds when s is varied from 5 to 25 for K=1024 bits. However, due to the expensive encryption costs, the computation cost of A varies from 207 to 538 milliseconds when s is changed from 5 to 25. On the other hand, due to the decryption costs in SMPA, the computation cost of T varies from 30 to 130 milliseconds when s is changed from 5 to 25. A similar trend can be observed for K=2048 bits as shown in Figure 3.5(d). In short, when $s = 5$, the computation costs of A, T, and U_j are 1.79, 0.24 and 0.002 seconds respectively. For any fixed parameters, the observation is that the computation costs of A and T are increased by a factor of almost 8 whereas U_j's time is increased by a factor of 2 when K is doubled.

Based on the above results, it is clear that the computation costs incurred due to Phase 2 are much higher than those of Phase 1. This further validates the computational analysis of PFR as discussed in the previous sub-section.

Furthermore, the total run time of PFR based on varying values of h and s is computed. That is, the total time took by the PFR protocol to compute the recommendation score corresponding to the potential user U_j (similar analysis can be deduced for other potential candidates) is computed. The number of child friends for each potential user (and A) is fixed to 50. The results are as shown in Figures 3.5(e) and 3.5(f). For K=1024, as shown in Figure 3.5(e), the total time does not change much when h is changed from 2 to 3 for both s =1 and 5. For example, when $s = 5$, the run time of PFR varies from 241.8 to 245.8 milliseconds when h is changed from 2 to 3. This is because the cost of Phase 1 does not change much since there is no need of SMP when $h = 2$ and 3. Also, the cost of Phase 2 almost remains the same for any fixed s (assuming m is also fixed). However, the total cost increases as h is varied from 3 to 6. For example, when $s = 5$, the total time for PFR to compute the recommendation score for U_j varies from 245.8 to 351.05 milliseconds when the value of h is changed from 3 to 6. A similar trend can be observed for $s = 1$ as shown in 3.5(e). Also, for any given value of h, the computation time of PFR is almost increased by a factor of 1 to 2 when s is changed from 1 to 5. E.g., when $h = 6$, the computation time of PFR varies from 264.25 to 351.05 milliseconds when s is changed from 1 to 5. In addition, observe that for any fixed values of h and s, the running time of PFR grows almost linearly with n. A similar trend can be observed for K=2048 as shown in Figure 3.5(f).

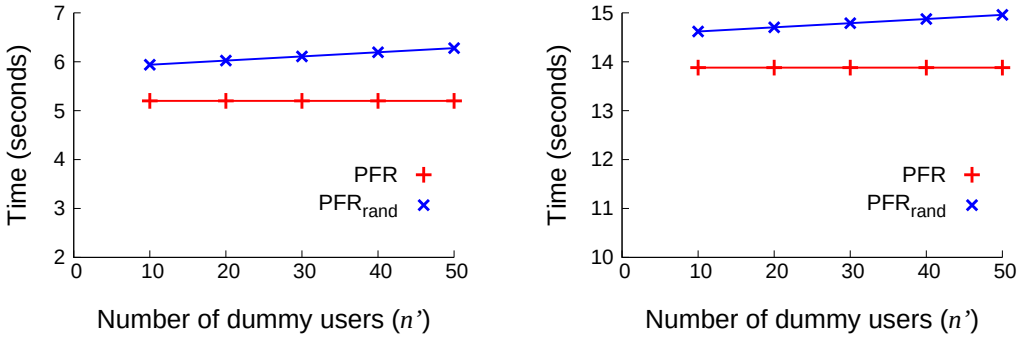
These results show that most of the significant computation (more than 97%) is from A and T. The computation cost incurred on all internal nodes is negligible. In addition, for A and T the computation time grows linearly with s and n. Furthermore,

when the size of encryption key doubles, the computation time of PFR increases by almost a factor of 8 for any fixed parameters.

3.8.3. Computation Costs - PFR Vs. $\mathbf{PFR_{rand}}$. In this sub-section, the computation costs of PFR_{rand} are compared with that of PFR. As mentioned earlier, remember that adding random dummy users to take participation in the PFR_{rand} protocol does not change the relative ordering among the recommendation scores of actual potential candidates. That is, the effectiveness of PFR_{rand} is the same as in PFR. For the rest of this sub-section, the Paillier key size K is fixed to 1024 bits (however, similar analysis can be deduced for K=2048 bits). The comparison results are as shown in Figure 3.6.

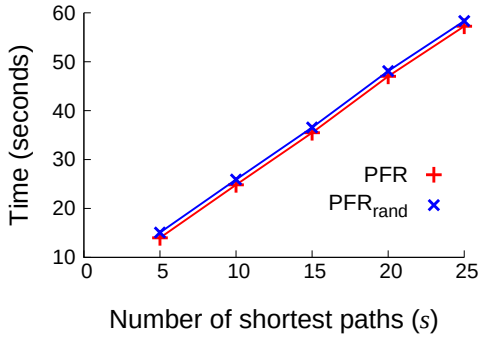
For any given parameters, it is important to note that the computation cost of Phase 1 in PFR_{rand} is the same as in PFR. This is because of the fact that though additional dummy child friends are added for each internal user (and A), he/she will operate on the encrypted partial data received from only his/her actual child friends. Therefore, first, the computation costs of Phase 2 in PFR and PFR_{rand} are compared.

Suppose the number of actual child friends for each potential candidate U_j be 50, i.e., $|Ch(U_j)| = 50$. Also, let $m = 50$, $n = 100$ and $h = 6$. Now, the computation time of Phase 2 in the proposed protocols for varying n' and $s = 1$ are evaluated, where n' denotes the total number of dummy random users participating in the PFR_{rand} protocol. As shown in Figure 3.6(a), the computation time of Phase 2 in PFR is 5.2 seconds and it remains constant with varying n' since Phase 2 is independent of n' in PFR. Whereas, the computation cost of Phase 2 in PFR_{rand} varies from 5.93 to 6.27 seconds when n' is varied from 10 to 50. It is clear that the extra cost incurred on Phase 2 in PFR_{rand} is not much compared to the cost of Phase 2 in PFR. A similar trend can be observed for $s = 5$ as shown in Figure 3.6(b). Briefly, the computation time of Phase 2 in PFR remains to be constant at 13.88



(a) Cost of Phase 2 for $s = 1$

(b) Cost of Phase 2 for $s = 5$



(c) Total cost for $n' = 50$

Figure 3.6: Comparison of computation costs of PFR_{rand} and PFR for K=1024

seconds whereas in PFR_{rand} the cost varies from 14.61 to 14.95 seconds when n' is varied from 10 to 50.

Also, the total run time of PFR and PFR_{rand} for varying s and by fixing n' to 50 are computed. The results are as shown in Figure 3.6(c). The total run time of PFR varies from 13.99 to 57.25 seconds whereas that of PFR_{rand} varies from 15.07 to 58.33 seconds when s is changed from 5 to 25. Irrespective of the value of s and for any fixed parameters, observe that the total run time of PFR is very close to that of PFR_{rand}.

Note that friend recommendations are generally performed on a daily basis (not a real-time application); therefore, the performances of the proposed protocols are reasonable in practice. The main advantage is that the proposed protocols make it

possible to do friend recommendations even in a privacy-preserving social networking environments.

3.9. CONCLUSION

Friend recommendation has become an important service in many online social networks to enhance the development of the entire network as well as to provide opportunities for users to expand their social relations. Due to privacy concerns [19, 21, 51], many online social networks are providing various privacy settings to users. Existing friend recommendation algorithms do not take privacy into account; therefore, they are not applicable in privacy-preserving social networking environments. This work first proposes a new two-phase private friend recommendation (PFR) algorithm based on the network structure as well as real messages exchanged between users. The proposed PFR protocol computes the recommendation scores of all users within a radius of h from the target user A by using the similarity metric proposed in [14] as a baseline. In particular, the proposed protocol generates the (scaled) recommendation scores along with the corresponding user IDs in such a way that the relative ordering among the users in the TOP-K list of recommended users is preserved (i.e., same accuracy as in [14]).

This work provided an in-depth security and complexity analysis of the proposed PFR protocol and also addressed various implementation details related to PFR in practice. In addition, this work demonstrated a new security issue in the current online social networks due to the inherent message flow information between different entities. To mitigate this issue or to provide better security, an extended version of the proposed protocol was developed using randomization technique. Also, this work showed the practical applicability of the proposed protocols through extensive experiments based on different parameters.

4. INTEREST-DRIVEN PRIVATE FRIEND RECOMMENDATION

Most of the time, users in a social network wish to find new friends based on their own interests. For instance, a biologist working on cancer cells may want to collaborate with new researchers who are also interested in the relevant topics. A computer science researcher may want to make new friends who are working in the field of "Social Networks". Due to social engineering attacks [24], a social network user may not want to keep his or her personal profile publicly available. Especially, considering the size of the social network users (which is usually in millions), finding new friends based on both social network structure and users' dynamic interest will make the private friend recommendation (PFR) problem even more challenging.

This section considers the set of users within a group G and proposes a set of solutions to privately recommend friends within this group based on users' specific interests. Note that the profile information of users even within a public group (where any one can join the group) can be treated as private and is not allowed to be visible to other users even within the same group. This work assumes that users' friend lists and social tags are private information. In other words, friendship between any two users is treated as sensitive information, and the social tags [59] attributed to any given user are also treated as private.

4.1. PROBLEM DEFINITION

Consider G as a group in a social network, and let n be the number of users in G. Without loss of generality, let $u_1, \ldots, u_n$ denote the users in G and T be the ontology tree constructed from the domain knowledge using the possible set of social tags in G. For the rest of this section, assume that T is constructed by the domain expert such as the network administrator (more details on how to construct T are

given in later part of this section). Each leaf node in T represents a set of social tags, and each internal node denotes a category C_k representing the generalization for the set of social tags under the sub-tree rooted at C_k. In addition, each user u_i's social tags in G is denoted by Tags(u_i), and let FL(u_i) denote the list of user IDs who are friends of u_i, for $1 \leq i \leq n$. In the literature, there exist several similarity metrics for computing the social closeness score (denoted as SC) between any two given users as part of the friend recommendation process [15]. Each metric has its pros and cons, and this work adopts the scoring function proposed in [13] that considers both social tagging information and topological structures in the social network. More specifically, the social closeness between two users u_i and u_j in G is defined as [13]:

$$\mathrm{SC}(u_i, u_j) = \theta \cdot \mathrm{NS}(u_i, u_j) + (1 - \theta) \cdot \mathrm{TS}(u_i, u_j) \tag{4.1}$$

where θ is a control parameter; $\mathrm{NS}(u_i, u_j)$ and $\mathrm{TS}(u_i, u_j)$ are the network and social tag similarity scores, between u_i and u_j, respectively defined as below:

$$\begin{aligned} \mathrm{NS}(u_i, u_j) &= \mathrm{Cosine}(u_i, u_j) = \frac{v_i \cdot v_j}{||v_i|| \cdot ||v_j||} \\ \mathrm{TS}(u_i, u_j) &= \frac{1}{\sum_{k,k+1 \ \in \ \mathrm{SP}(\mathrm{Cat}(u_i),\mathrm{Cat}(u_j))} \frac{2}{d(k)+d(k+1)}} \end{aligned}$$

In the above equations, $\mathrm{Cosine}(u_i, u_j)$ denotes the cosine similarity between the friendship vectors v_i and v_j of u_i and u_j respectively. v_i and v_j are constructed from FL(u_i) and FL(u_j), such that $|v_i| = |v_j| = n$ (denoting the size of global friend list space of G) and $v_i[k] = 1$ if $u_k \in \mathrm{FL}(u_i)$, otherwise $v_i[k] = 0$, for $1 \leq k \leq n$. $||v_i||$ and $||v_j||$ denote the Euclidean norms of v_i and v_j. Cat(u_i) and Cat(u_j) denote the category (deepest non-leaf) nodes of u_i and u_j. SP(Cat(u_i), Cat(u_j)) denotes the trail of nodes in the shortest path between nodes Cat(u_i) and Cat(u_j). Nodes k and $k+1$ are two

consecutive nodes in the shortest path, and $d(k)$ is the depth of node k. More details about how to compute the SC score are given later along with a concrete example.

Given a snapshot of the social network for users in G along with their social tags and the ontology tree T, the problem of computing social closeness score between a target user u_i and any other user u_j based on Equation 4.1 is straightforward. However, the problem becomes non-trivial if the network structure and social tags are considered as users' private information. More formally, this work considers the following users' profile information as private:

- Friendship between any two users u_i and u_j should not be revealed to any other user. This further implies that $\text{FL}(u_i)$ is only known to u_i, for $1 \leq i \leq n$.
- The social tags attributed to u_i (i.e., $\text{Tags}(u_i)$) are only known to u_i.

Without loss of generality, let $G_i = \langle u_{j_1}, \ldots, u_{j_t} \rangle$ denote a subset of users in G (excluding u_i and friends of u_i). Here G_i is chosen by u_i either based on his/her interest or randomly. More details regarding this issue are provided in later sections. According to the above privacy assumptions, the goal of this work is to securely compute the social closeness scores between a target user u_i and each user in G_i in a privacy-preserving manner. As mentioned earlier, this process is termed as private friend recommendation (PFR). Formally, the PFR problem is defined as:

$$\text{PFR}(u_i, G_i \subset G) \rightarrow \langle \text{SC}(u_i, u_{j_1}), \ldots, \text{SC}(u_i, u_{j_t}) \rangle \tag{4.2}$$

At the end of the PFR protocol, only u_i knows the values of $\text{SC}(u_i, u_{j_l})$, for $1 \leq l \leq t$. Once the t social closeness scores are known to u_i, he/she can either send friend requests to or browse through the pubic profiles of Top-K users (i.e., users in G_i with Top-K social closeness scores with u_i) and take action accordingly.

4.2. MAIN CONTRIBUTIONS

This work proposes two new PFR protocols based on both the network structure and users' social tags using a knowledge based ontology tree. Both of the proposed protocols compute the social closeness scores, according to Equation 4.1, between a target user u_i in a group G (who wish to make new friends) and each user in a subset $G_i \subset G$ without revealing any private information between each other. The first protocol assumes $\mathrm{FL}(u_i)$ and $\mathrm{FL}(u_{j_l})$ as private information and the social tags of users can be revealed to the network administrator, and returns $\mathrm{SC}(u_i, u_{j_l})$ without disclosing $\mathrm{FL}(u_i)$ to u_{j_l} and $\mathrm{FL}(u_{j_l})$ to u_i, for $u_{j_l} \in G_i$. Whereas the second protocol assumes not only $\mathrm{FL}(u_i)$ and $\mathrm{FL}(u_{j_l})$ are private but also their social tags are private. At the end of both protocols, only user u_i knows the $\mathrm{SC}(u_i, u_{j_l})$ scores (based on Equation 4.1) and decides whether to send a friend request to user u_{j_l}, for $u_{j_l} \in G_i$. Specifically, the main contributions of this work are summarized below:

- *Interest-Driven.* The first protocol provides more flexibility to users in terms of identifying new friends based on his/her specific interest (i.e., choosing users in G_i based on his/her own interest).

- *Security.* The second protocol preserves the privacy of individuals (i.e., both users' friend lists and social tags) and is more secure than the first protocol. The security guarantee follows the well-known semi-honest security definition of secure multiparty computation [38, 39, 50].

- *Trade-off.* Since some of the users' computations in the first protocol are pushed to the network administrator, the first protocol is more efficient than the second protocol. However, this efficiency gain comes at the expense of releasing the users' social tags directly to the network administrator. Therefore, the proposed protocols act as a trade-off between efficiency and security.

4.3. RELATED WORK

Social network analyses have been utilized for various business applications [6], such as predicting the future [28] and developing recommendation systems [31, 32]. With growing interest of expanding a person's social circle, friend recommendation has become an important service in many online social networks. Along this direction, Silva et al. [11] proposed a novel user calibration procedure based on a genetic algorithm to optimize the three indices derived from the structural properties of social networks. Xie [12] designed a general friend recommendation framework to recommend friends based on the common interests by characterizing user interests in two dimensions - context (e.g., location and time) and content. By treating friend recommendation process as a filtering problem, Naruchitparames et al. [10] developed a two-step approach to provide quality friend recommendations by combining cognitive theory with a Pareto-optimal genetic algorithm. As an independent work, Gou et al. [13] developed a visualization tool (named as SFViz) that allows users to explore for a potential friend with an interest context in social networks. Their method considers both semantic structure in social tags and topological structures in social networks to recommend new friends. Nevertheless, Facebook uses the "People You May Know" feature to recommend friends based on the simple "friend-of-a-friend" approach [34].

Due to various privacy issues [17, 18, 19, 20, 21, 22, 23], many users keep their profile information as private. Existing friend recommendation techniques [9, 10, 11, 12, 13, 14] do not take users' privacy into consideration; therefore, they cannot be directly applied. Only recently, researchers have focused on developing accurate and efficient PFR methods. Along this direction, Dong et al. [35] proposed a method to securely compute and verify social proximity between two users using cosine similarity in mobile social networks. This work differs from theirs in two aspects. First, the problem setting is entirely different from theirs. This work considers the users in an

(Internet-based) online social network group G, where the friend list and social tags of each user are treated as private. Whereas, in their approach, (mobile social network) users physical location is treated as private. Secondly, the proposed protocols in this section identify new friends based on the users' friend lists and their social tags. More specifically, this work uses the scoring function proposed in [13] to measure the social closeness between any two given users. On the other hand, their approach identifies new friends who happen to be in the physical vicinity of the target user. That is, social coordinates (users geographical location) are used to compute the social proximity between users.

Machanavajjhala et al. [36] formally analyzed the trade-offs between accuracy and privacy of private friend recommendations using differential privacy [37]. In their work, the authors used the existing differentially private algorithms as underlying sub-routines and assumed the existence of PFR protocols based on these sub-routines. They have considered network topology based similarity metrics. Different from their work, the algorithm proposed in this section securely computes the social closeness score using the similarity scoring function proposed in [13] which is based on the social network topology as well as the social tagging information. Also, according to their claims, if privacy is to be preserved when using the common neighbors utility function [15], only users with $\Omega(\log n)$ friends can hope to receive accurate recommendations, where n is the number of users in the graph. Furthermore, the users' privacy in [36] is based on differential privacy. Whereas, the privacy guarantees in this work are based on an entirely different security model, namely the semi-honest security definitions from the field of secure multiparty computation (SMC) [38, 39]. Under the SMC model, this work develops accurate PFR protocols. In particular, the second proposed protocol recommends friends accurately, similar to [13], and simultaneously preserves the privacy of each user (i.e., users' friend lists as well as social tags are protected).

4.4. PRELIMINARIES

This sub-section highlights the steps involved in computing the category for a given user $u_i \in G$ using the social tags of u_i and T. Then, details on how to compute the cosine similarity score between friend lists of users are given. Also, a running example for computing the social closeness score based on Equation 4.1 is presented. Finally, this sub-section describes the properties exhibited by additive homomorphic encryption schemes. Notations commonly used throughout this section are given in Table 4.1.

4.4.1. Computation of User Category. Without loss of generality, let $C_1, \ldots, C_m$ be the deepest non-leaf category nodes (in order from left to right) in T. For any given user $u_i \in G$, assign u_i to exactly one of the category nodes, which is referred to as "category of user u_i" and is denoted by $\text{Cat}(u_i)$, as follows[13]:

- Compute the matching score (MS) between u_i's social tags and each category node C_x as given by

$$\text{MS}(u_i, C_x) = \frac{\sum_{h \in \text{Tags}(u_i) \cap \text{allTags}(C_x)} f(h) \cdot d(h)}{\sum_{y \in \text{allTags}(C_x)} f(y)} \tag{4.3}$$

  where $f(h)$ and $d(h)$ denote the frequency and depth of tag h respectively; h is common to $\text{Tags}(u_i)$ and $\text{allTags}(C_x)$, where $\text{Tags}(u_i)$ returns all tags of user u_i, and $\text{allTags}(C_x)$ returns tags under category C_x and all ancestors of C_x (omitting root node as stated in[13]).

- Next, pick the largest score and assign user u_i to the corresponding category (i.e., $\text{Cat}(u_i)$).

Example 1. Consider the sample ontology tree T for a group G as shown in Figure 4.1. Following from the Figure 4.1, all tags related to category C_3 are given as $\text{allTags}(C_3) = \{t_1, t_2, t_3, C_3, C_{1,3}\}$. Without loss of generality, let *Bob* be a user in G

Table 4.1: Some common notations used in the interest based PFR protocols

HPEnc	An Additive Homomorphic Probabilistic Encryption system
G	A group in social network with users $u_1, \ldots, u_n$
T	Ontology tree (generated by the network administrator)
$\langle E_{pk}, D_{pr} \rangle$	A pair of HPEnc based encryption and decryption function with (pk, pr) as the corresponding public-private key pair
$\text{FL}(u_i)$	Friend list of user u_i
$\text{Tags}(u_i)$	Social tags of user u_i
$\text{Cat}(u_i)$	Category of user u_i
s	Scaling factor

such that $\text{Tags}(Bob) = \{t_1, t_3, t_9, C_3\}$. The common set of tags between C_3 and Bob are $\{t_1, t_3, C_3\}$. In addition, suppose the frequencies are as follows. $f(t_1) = 2, f(t_2) = 5, f(t_3) = 1, f(C_3) = 4$, and $f(C_{1,3}) = 2$. The matching score between C_3 and Bob, based on Equation 4.3, is computed as follows:

$$\begin{aligned}\text{MS}(Bob, C_3) &= \frac{f(t_1) \cdot d(t_1) + f(t_3) \cdot d(t_3) + f(C_3) \cdot d(C_3)}{f(t_1) + f(t_2) + f(t_3) + f(C_3) + f(C_{1,3})} \\ &= \frac{2 \cdot 3 + 1 \cdot 3 + 4 \cdot 2}{2 + 5 + 1 + 4 + 2} \\ &= \frac{17}{14}\end{aligned}$$

4.4.2. Computation of Cosine Similarity Score. For any given user $u_i \in G$, let v_i denote the friendship vector of u_i derived from the global user space of G. More formally, the vector v_i is defined as follows:

$$v_i[k] = \begin{cases} 1 & \text{if } u_k \in \text{FL}(u_i) \\ 0 & \text{otherwise} \end{cases}$$

That is, if $v_i[k] = 1$, then the user corresponding to the k^{th} dimension in the global user space is a friend of u_i. Without loss of generality, let the global user space be

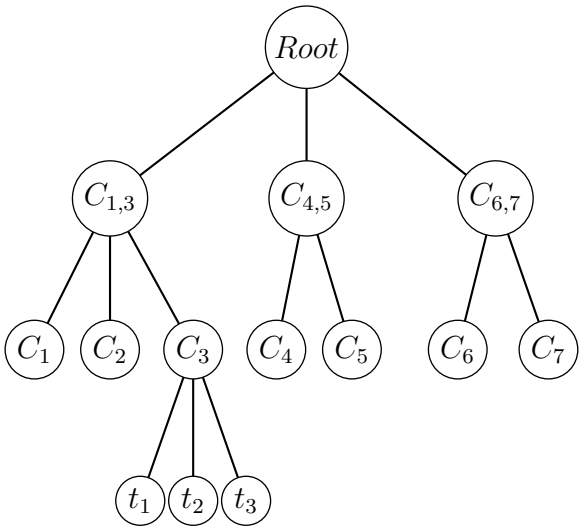


Figure 4.1: A sample ontology tree T along with the tags of C_3

$u_1, \ldots, u_n$ (in order). If $v_i[k] = 1$, then u_i and u_k are friends. Note that if u_i is not a friend of u_k, then the entry corresponding to the k^{th} dimension (i.e., $v_i[k]$) is zero.

Once the users' friend lists are represented as friendship vectors, cosine similarity between the friend lists of any two users u_i and u_j gives the cosine of the angles between the corresponding two friendship vectors v_i and v_j. The following equation captures the cosine similarity score between u_i and u_j.

$$\text{Cosine}(u_i, u_j) = \frac{\sum_{k=1}^{n} v_i[k] \cdot v_j[k]}{||v_i|| \cdot ||v_j||} = \vec{v}_i \bullet \vec{v}_j \tag{4.4}$$

where $||v_i||$ and $||v_j||$ denote the Euclidean norms* of v_i and v_j; $\vec{v}_i$ and $\vec{v}_j$ are the normalized vectors of v_i and v_j respectively, and $\vec{v}_i \bullet \vec{v}_j$ denotes the dot product between $\vec{v}_i$ and $\vec{v}_j$. Note that the value of cosine similarity score always varies between 0 and 1. A cosine similarity score of zero means the two friendship vectors are orthogonal, and the two users have no friends in common. On the other hand, a cosine similarity score of one means that the friend lists of both users are the same.

*More specifically, the Euclidean length/norm of v_i and v_j are given as $||v_i|| = \sqrt{\sum_{k=1}^{n} v_i[k]^2}$ and $||v_j|| = \sqrt{\sum_{k=1}^{n} v_j[k]^2}$

Example 2. Consider two users Bob and $Charles$ in G with friend lists $\{Ellis, Reed, Beck, Walton\}$ and $\{Beck, Steele, Bush, Walton, Hodges\}$ respectively. Assume that $\{Ellis, Beck, Bob, Francis, Walton, Steele, Bush, Joseph, Reed, Charles, Hodges\}$ is the global user space of G. Observe that $Francis$ and $Joseph$ are neither friends of Bob nor $Charles$. The friendship vectors for both Bob and $Charles$ are given as $v_{Bob} = \langle 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0\rangle$ and $v_{Charles} = \langle 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1\rangle$ respectively. In addition, $||v_{Bob}|| = 2$ and $||v_{Charles}|| = \sqrt{5}$. Therefore, the cosine similarity between the friendship vectors of Bob and $Charles$, based on Equation 4.4, is derived as follows:

$$\begin{aligned}\text{Cosine}(Bob, Charles) &= \frac{\sum_{k=1}^{11} v_{Bob}[k] \cdot v_{Charles}[k]}{||v_{Bob}|| \cdot ||v_{Charles}||} \\ &= \frac{1}{\sqrt{5}}\end{aligned}$$

4.4.3. Computation of Social Closeness Score. Following from the above discussion, consider the scenario of computing the social closeness score between Bob and $Charles$ in G based on Equation 4.1. Let C_3 and C_4 be the categories of Bob and $Charles$ respectively. Since $\text{NS}(Bob, Charles) = \text{Cosine}(Bob, Charles) = \frac{1}{\sqrt{5}}$, let us compute the tag similarity score between Bob and $Charles$. As mentioned earlier, the tag similarity score between Bob and $Charles$ is given by

$$\text{TS}(Bob, Charles) = \frac{1}{\sum_{k,k+1 \ \in \ \text{SP}(C_3, C_4)} \frac{2}{d(k)+d(k+1)}}$$

where $\text{SP}(C_3, C_4)$ denotes the trail of nodes on the shortest path between C_3 and C_4, node k and $k+1$ are the consecutive nodes on the shortest path, and $d(k)$ denotes the depth of node k in T. From Figure 4.1, $\text{SP}(C_3, C_4) = \{C_3, C_{1,3}, Root, C_{4,5}, C_4\}$. In addition, $d(C_3) = d(C_4) = 2$, $d(C_{1,3}) = d(C_{4,5}) = 1$, and $d(root) = 0$. Therefore,

by substituting these values in the above formula, the tag similarity score is given by:

$$\begin{aligned}\mathrm{TS}(Bob, Charles) &= \frac{1}{\frac{2}{3} + 2 + 2 + \frac{2}{3}} \\ &= \frac{3}{16}\end{aligned}$$

Suppose the value of control parameter θ be 0.75. Then, the social closeness score between Bob and $Charles$, based on Equation 4.1, is given as:

$$\begin{aligned}\mathrm{SC}(Bob, Charles) &= 0.75 \cdot \mathrm{NS}(Bob, Charles) \\ &\quad + (1 - 0.75) \cdot \mathrm{TS}(Bob, Charles) \\ &= 0.75 \cdot 0.44 + 0.25 \cdot 0.18 \\ &= 0.375\end{aligned}$$

4.4.4. Additive Homomorphic Probabilistic Encryption. The proposed protocols utilize an additive homomorphic encryption scheme which is probabilistic in nature. In the literature, such a scheme is referred to as HPEnc system (such as Paillier cryptosystem [55]). Let E_{pk} and D_{pr} be the encryption and decryption functions of an HPEnc system with pk and pr as their public and private keys respectively. In addition, let N be the RSA modulus generated by taking the product of two large primes of similar bit-length. For any given plaintext messages a, b, and $c \in \mathbb{Z}_N$, the HPEnc system exhibits the following properties:

- **Homomorphic Addition -** $E_{pk}(a + b) = E_{pk}(a) \cdot E_{pk}(b) \bmod N^2$
- **Scalar Multiplication -** $E_{pk}(c \cdot a) = E_{pk}(a)^c \bmod N^2$
- **Semantic Security -** Given a set of ciphertexts and public key pk, it is not feasible for a computationally bounded adversary to derive any information about the plaintexts [49, 56].

4.5. THE PROPOSED PROTOCOLS

This sub-section presents the proposed PFR protocols which compute the social closeness score between a target user u_i (who wish to make new friends) and each user in G_i in a privacy-preserving manner. Both the proposed protocols aim at protecting the users' friend lists and social tags. At the end of the proposed protocols, only user u_i knows the social closeness scores and decides whether to send a new friend request (such as "Add Friend" in Facebook) to users in G_i. This work explicitly considers the following set of assumptions for the rest of this sub-section:

- User u_i owns E_{pk} and D_{pr}: an additive homomorphic encryption scheme (e.g., Paillier cryptosystem [55]) with a public-private key pair (pk, pr) and N as the ring size.
- Each user in G is aware of all the other users within group G which is practical since it is indeed the case in many online social networks. However, the friend list and social tags of each user are treated as private information.

4.5.1. The PFR$_\alpha$ Protocol. This protocol assumes that users' social tags can be revealed to the network administrator to reduce the work load of users. Though this protocol releases users' social tags to the network administrator for efficiency reasons, users' friend lists are kept as private. In addition, assume that the network administrator acts as a third party and there is no collusion between users and the network administrator. At the end of the PFR$_\alpha$ protocol, the final social closeness scores are revealed only to u_i. Under the above assumptions, the PFR$_\alpha$ protocol consists of the following two stages:

Stage 1 - Computation of Users' Categories: During this Stage, the network administrator first computes ontology tree T and assigns each user $u_k \in G$ to one of the deepest non-leaf nodes as the category of u_k (denoted as Cat(u_k)), and updates

T by attaching the user ID of u_k to the node $\text{Cat}(u_k)$. At the end of this stage, the updated T is published to the users in G.

Stage 2 - Secure Computation of Social Closeness Scores: User u_i first picks a list of users $G_i \subset G$ (omitting u_i and friends of u_i) based on his/her interest using T resulted from Stage 1. After this, u_i securely computes the social closeness scores with each user in G_i.

The overall steps involved in the PFR_α protocol are highlighted in Algorithm 4. Briefly, during Stage 1, each user sends his/her social tags to the network administrator. Upon receiving the tags, the network administrator computes the category for each user based on Equation 4.3, and assigns each user ID to his/her category by updating T. After this, the network administrator publishes the updated T to the users in G. During Stage 2, target user u_i first selects a subset of users in G (i.e., G_i) based on his/her own interests. For example, if u_i is interested in the users under category C_j, then he/she can compute the social closeness scores with only the users in C_j based on Equation 4.1. To be more generic, this work assumes that G_i can contain users from multiple categories. For each user u_{j_l} in G_i, u_i can compute $\text{TS}(u_i, u_{j_l})$ locally since he/she knows (from the published T in Stage 1) where node $\text{Cat}(u_{j_l})$ (i.e., the category node for user u_{j_l}) is located in T. However, to compute $\text{NS}(u_i, u_{j_l})$, one need to calculate the cosine similarity score between u_i and u_{j_l} securely such that $\text{FL}(u_i)$ is not revealed to u_{j_l} and $\text{FL}(u_{j_l})$ is not revealed to u_i. This can be seen as a sub-problem of secure similar document detection (SSDD). Therefore, existing SSDD techniques can be utilized to solve the problem of secure computation of cosine similarity [60, 61]. Finally, for each $u_{j_l} \in G_i$, u_i combines $\text{NS}(u_i, u_{j_l})$ and $\text{TS}(u_i, u_{j_l})$ based on the linear operation in Equation 4.1 to derive the social closeness score $\text{SC}(u_i, u_{j_l})$, for $1 \leq l \leq t$.

Algorithm 4 $\text{PFR}_\alpha(u_i, G_i \subset G) \rightarrow \langle \text{SC}(u_i, u_{j_1}), \ldots, \text{SC}(u_i, u_{j_t}) \rangle$

Require: pr is known only to u_i, pk is public, and $\text{FL}(u_k)$ are private $\forall\ u_k \in G$

1: User u_k, **for** $1 \leq k \leq n$ **do**:

(a). Send $\text{Tags}(u_k)$ to the network administrator

2: Network administrator:

(a). Receive $\text{Tags}(u_k)$ from u_k, for $1 \leq k \leq n$

(b). Generate T

(c). **for each** $u_k \in G$ **do**:

- Compute $\text{Cat}(u_k)$ based on Equation 4.3

(d). Update T and publish it in G.

3: User u_i:

(a). Select G_i using T

(b). Compute $\text{TS}(u_i, u_{j_l})$, for $l = 1, \ldots, t$

(c). **for** $k = 1$ to n **do**:

- $\vec{v}_i[k] \leftarrow \frac{v_i[k]}{||v_i||}$
- $V_i[k] \leftarrow E_{pk}(s \cdot \vec{v}_i[k])$, where s is a scaling factor

(d). Send V_i to user u_{j_l}, for $l = 1, \ldots, t$

4: User u_{j_l}, **for** $1 \leq l \leq t$ **do**:

(a). Receive V_i from u_i

(b). $\vec{v}_{j_l}[k] \leftarrow \frac{v_{j_l}[k]}{||v_{j_l}||}$, for $1 \leq k \leq n$

(c). $Z_{j_l} \leftarrow \prod_{k=1}^{n} V_i[k]^{s \cdot \vec{v}_{j_l}[k]} \bmod N^2$; send Z_{j_l} to user u_i

5: User u_i:

(a). **for** $1 \leq l \leq t$ **do**:

- Receive Z_{j_l} from u_{j_l}
- Decryption: $\Phi(u_i, u_{j_l}) \leftarrow D_{pr}(Z_{j_l})$
- $\text{SC}(u_i, u_{j_l}) \leftarrow \theta \cdot \frac{\Phi(u_i, u_{j_l})}{s^2} + (1 - \theta) \cdot \text{TS}(u_i, u_{j_l})$

Note that the control parameter θ in Equation 4.1 can be either a static value (published by the network administrator) or chosen by the target user u_i. Now, the steps involved in each of the two stages are explained in detail.

Stage 1 - Computation of Users' Categories. The overall steps for Stage 1 in PFR_α are highlighted as steps 1 and 2 in Algorithm 4. Initially, each user u_k sends his/her social tags (i.e., $\text{Tags}(u_k)$) to the network administrator, for $1 \leq k \leq n$. Note that since this work assumes that there is no collusion between the network administrator and the users, it implies that $\text{Tags}(u_k)$ is not leaked to users other than u_k. Upon receiving the social tags, an ontology tree T is constructed by the network administrator, such that the leaf nodes in T are individual tags and the internal nodes (i.e., non-leaf nodes) can be regarded as categories denoting the generalization of all social tags under the sub-tree rooted at the internal node [13, 62]. Since social tags can be represented as XML data, T can be generated automatically as long as the network administrator knows the frequencies of individual tags.

Once T is built, the network administrator computes the matching score (MS) between each u_k and all deepest non-leaf nodes and assigns the category that have the highest MS score to u_k as the category node for u_k, for $1 \leq k \leq n$. Once the users are assigned to each category, this information (i.e., update T by including user IDs to their respective categories in T) is published within the group.

Stage 2 - Secure Computation of Social Closeness Scores. Following from Stage 1, the leaf nodes in T (published by the network administrator) represent the user IDs and the deepest internal nodes represent the category label for all the leaf nodes under it. Suppose user u_i wants to compute the social closeness scores with t users in the group $G_i \subset G$ denoted by $u_{j_1}, \ldots, u_{j_t}$. The observation is that since u_i knows the categories of all users in G, he/she can choose the potential users based on his/her own interests. For example, consider a biology community where users can be assigned to multiple research categories in T. If u_i is interested only in researchers

working on cancer cells, he/she can choose only the user IDs under the cancer cell category and proceed further. Here the users in G_i may belong to multiple categories and the choice purely depends on the u_i's interest. The main steps involved in Stage 2 of PFR$_\alpha$ are shown as steps 3, 4, and 5 in Algorithm 4.

Once the group G_i is chosen, user u_i first computes TS(u_i, u_{j_l}) locally, for $1 \leq l \leq t$. After this, he/she computes $\vec{v}_i$ (using FL(u_i)) and encrypts the scaled $\vec{v}_i$ component-wise. Note that since the encryption scheme operates on group $\mathbb{Z}_N$ (i.e., integer-based) and as $\vec{v}_i[k]$ can be fractional, one need to convert $\vec{v}_i[k]$ to integer before encrypting it, for $1 \leq k \leq n$. This is the reason for multiplying each component of $\vec{v}_i$ (i.e., $\vec{v}_i[k]$) with the scaling factor s before encrypting it. In general, the value of s can be either static (chosen by the network administrator) or dynamic (varies with target user u_i). For example, when $\vec{v}_i[k] = 0.143$ (3-point precision), a scaling factor (i.e., s) of 1000 is sufficient. Briefly, u_i computes $V_i[k] = E_{pk}(s \cdot \vec{v}_i[k])$, for $1 \leq k \leq n$, and sends V_i to each user u_{j_l}, for $1 \leq l \leq t$. Note that if $s \cdot \vec{v}_i[k]$ is still a floating-point number, then the ceiling value of it can be taken before encryption. For the rest of this section, the proposed protocols simply omit the ceiling operation.

Upon receiving V_i from u_i, each user u_{j_l} performs the following operations:

- Compute the friendship vector v_{j_l} from FL(u_{j_l}) and normalize it to get $\vec{v}_{j_l}$. That is, for $1 \leq k \leq n$:

$$\vec{v}_{j_l}[k] = \frac{v_{j_l}[k]}{||v_{j_l}||}$$

- Compute the encrypted cosine similarity score as $Z_{j_l} = \prod_{k=1}^{n} V_i[k]^{s \cdot \vec{v}_{j_l}[k]} \bmod N^2$ and send Z_{j_l} to user u_i. Note that the exponent term should be an integer in $\mathbb{Z}_N$; therefore, u_{j_l} first multiplies $\vec{v}_{j_l}$ by s component-wise (scaling) and then uses the result $s \cdot \vec{v_{j_l}}[k]$ (which is an integer in $\mathbb{Z}_N$) to do exponentiation operations.

Finally, u_i performs the following operations to compute the final social closeness scores for each u_{j_l}, where $1 \leq l \leq t$:

- Receive Z_{j_l} from user u_{j_l} and decrypt it to get $\Phi(u_i, u_{j_l}) = D_{pr}(Z_{j_l})$.

- Observe that:

$$\begin{aligned}\Phi(u_i, u_{j_l}) &= \sum_{k=1}^{n}(s \cdot \vec{v}_i[k]) \cdot (s \cdot \vec{v}_{j_l}[k]) \\ &= s^2 \cdot (\vec{v}_i \bullet \vec{v}_{j_l}) \\ &= s^2 \cdot \mathrm{NS}(u_i, u_{j_l})\end{aligned}$$

- Compute $\mathrm{SC}(u_i, u_j)$ using $\Phi(u_i, u_{j_l})$ and $\mathrm{TS}(u_i, u_{j_l})$ as follows.

$$\begin{aligned}\mathrm{SC}(u_i, u_{j_l}) &= \theta \cdot \frac{\Phi(u_i, u_{j_l})}{s^2} + (1-\theta) \cdot \mathrm{TS}(u_i, u_{j_l}) \\ &= \theta \cdot \mathrm{NS}(u_i, u_{j_l}) + (1-\theta) \cdot \mathrm{TS}(u_i, u_{j_l})\end{aligned}$$

4.5.2. Complexity Analysis of PFR$_\alpha$. Next, the computation and communication costs of the PFR$_\alpha$ protocol are analyzed.

Computation Cost. During Stage 1, users in G simply sends their social tags to the network administrator; therefore, there is no computation involved on the user side. The main computation cost of PFR$_\alpha$ occurs in Stage 2. During Stage 2, the target user u_i creates his/her friendship vector, normalizes and encrypts it component-wise (from step 3(c) of Algorithm 4). In addition, u_i has to perform t number of decryptions (from step 5(a) of Algorithm 4). However, in practice, the value of t is less than n and the time for encryption is almost the same as the decryption operation under Paillier cryptosystem [55]. Therefore, the computation cost of u_i is bounded by $O(n)$ encryptions, where n is the number of users or size of the global user space in G.

Furthermore, only the users in G_i participate in Stage 2 of PFR$_\alpha$. Since each user $u_{j_l} \in G_i$ receives a vector of size n from the target user u_i, the number

of exponentiation operations performed by u_{j_l} is bounded by O(n), at step 4(c) of Algorithm 4. On the other hand, the total computation cost by all users in G_i is bounded by O($t \cdot n$) exponentiations. In general, an encryption is much costlier than an exponentiation operation; therefore, the computation cost of u_i is more significant compared to u_{j_l}.

Communication Cost. In the PFR$_\alpha$ protocol, the communication occurs between each user and the network administrator (from Stage 1). In addition, the communication also happens between u_i and each user in G_i (from Stage 2). During Stage 1, each user $u_k \in G$ sends his/her social tags to the network administrator (following from step 1(a) of Algorithm 4). Therefore, the communication cost between each user u_k and the network administrator is bounded by $O(|\text{Tags}(u_k)|)$ in bits.

In addition, during Stage 2, the target user u_i sends his/her encrypted vector which is of size n to each user $u_{j_l} \in G_i$ (following from step 3(d) of Algorithm 4). Hence, the communication cost between u_i and u_{j_l} is bounded by $O(z \cdot n)$ in bits, where z denotes the encryption key size in bits.

4.5.3. Security Analysis of PFR$_\alpha$. The PFR$_\alpha$ protocol assumes that the users' friend lists are private. In addition, the social tags of one user are not revealed to any other user. One issue with the PFR$_\alpha$ protocol is the assumption of revealing social tags to the network administrator. Though the social tags of one user are not revealed to the other user, the above issue may not be allowed in some cases where users wish to hide their tags even from the network administrator.

Also, the category information of each user is published publicly within the group in order to compute TS(u_i, u_{j_l}), for $1 \leq l \leq t$. However, revealing the category of a user may leak valuable information as it is possible to extract the semantic relations by analyzing the possible set of tags under this category. Therefore, to achieve better protection, it is desirable to move the computation of matching scores

(i.e., computation of users' categories) from the network administrator to individual users giving rise to the second proposed protocol as explained in the next sub-section.

4.5.4. The PFR$_\beta$ Protocol. In this sub-section, in addition to the friend lists, assume that users' social tags as well as their category information are private. Along this direction, a new PFR protocol (denoted as PFR$_\beta$) is proposed. The basic idea is to allow users to have control over their data which is achieved by performing the data critical computations locally and the remaining non-local operations are performed on the encrypted data using the additive homomorphic properties as discussed earlier.

Since the users cannot directly compute their categories locally, PFR$_\beta$ explicitly includes an initialization step to compute T including the frequency details of each tag globally as follows. In the PFR$_\beta$ protocol, unlike PFR$_\alpha$, the users of G do not send their social tags directly to the network administrator. Instead, one of the user acts as a coordinator (say u_n) and each user sends his/her encrypted social tags (using the public key of the network administrator) to u_n. Then, u_n appends his/her own encrypted social tags to the list and permutes the global list of encrypted social tags, and sends it to the network administrator. After receiving, the network administrator decrypts each entry and constructs T as mentioned in the PFR$_\alpha$ protocol. However, in addition to the tags as leaf nodes, the network administrator also includes its global frequency information (i.e., each leaf node in T looks like $\langle \omega, f(\omega) \rangle$, where ω is a social tag). After this, the network administrator publishes T to the users in G. Since u_n permutes the global list of encrypted social tags, the network administrator cannot trace back which social tag belongs to which user.

Apart from the above initialization step, the PFR$_\beta$ protocol consists of the following two stages similar to PFR$_\alpha$. However, the steps involved in each stage of PFR$_\beta$ are different from that of PFR$_\alpha$.

Stage 1 - Computation of Users' Categories: During Stage 1, each user locally computes the category corresponding to him/her using the ontology tree T published by the network administrator from the initialization step.

Stage 2 - Secure Computation of Social Closeness Scores: The target user u_i randomly chooses t users (denoted as set $G_i \subset G$) and computes the social closeness scores with each user in G_i. At the end of this step, only u_i knows the scores and nothing is revealed to either u_i or other users in G_i.

The overall steps in PFR_β are highlighted in Algorithm 5. Now, the steps involved in each stage of the PFR_β protocol are discussed as below.

Stage 1 - Computation of Users' Categories. Unlike in PFR_α, the social tags of users in G are not disclosed to the network administrator in PFR_β. Instead, assume that the network administrator simply publishes the ontology tree T (where the leaf nodes consist of social tags and their frequencies) and makes it available to users in G. From T, each user $u_k \in G$ can locally compute the matching scores between $\text{Tags}(u_k)$ and tags of each category (deepest non-leaf nodes) and identifies the one with largest matching score as his or her corresponding category $\text{Cat}(u_k)$. Since the computation is done locally and as $\text{Tags}(u_k)$ is only known to user u_k, the category $\text{Cat}(u_k)$ is only known to user u_k. Thus, social tags of users as well as his/her category information are kept private. Stage 1 is shown as step 1 in Algorithm 5.

Stage 2 - Secure Computation of Social Closeness Scores. Following from Stage 1, user u_i knows only his/her category $\text{Cat}(u_i)$ (similarly, other users in G knows their respective categories). When user u_i wants to make new friends from a group of users $G_i \subset G$ (note that G_i is chosen randomly after excluding u_i and friends of u_i from G), u_i can securely compute the cosine similarity with each of them as discussed in the PFR_α protocol. That is, u_i computes the component-wise encryption of the scaled normalized friend list vector as $V_i[k] = E_{pk}(s \cdot \vec{v}_i[k])$, for $1 \leq k \leq n$. However, for any

Algorithm 5 $\text{PFR}_{\beta}(u_i, G_i \subset G) \rightarrow \langle \text{SC}(u_i, u_{j_1}), \ldots, \text{SC}(u_i, u_{j_t}) \rangle$

Require: pr is known only to u_i; s, pk, θ, and T are public (note that T is resulted from the initialization step); $\text{FL}(u_k)$, $\text{Tags}(u_k)$, and $\text{Cat}(u_k)$ are private $\forall\ u_k \in G$

1: User u_k, **for** $1 \leq k \leq n$ **do**:

(a). Compute u_k's category $\text{Cat}(u_k)$

2: User u_i:

(a). Select G_i randomly from G

(b). **for** $k = 1$ to n **do**:

- $\vec{v}_i[k] \leftarrow \frac{v_i[k]}{||v_i||}$
- $V_i[k] \leftarrow E_{pk}(s \cdot \vec{v}_i[k])$

(c). **for** $x = 1$ to m **do** (where m denotes the number of deepest non-leaf category nodes)

- Compute $\text{TS}(u_i, u'_x)$ (where u'_x is a dummy user such that $\text{Cat}(u'_x) = C_x$)
- $Z_i[x] \leftarrow E_{pk}(s^2 \cdot \text{TS}(u_i, u'_x))$

(d). Send V_i and Z_i to each user in G_i

3: Each user $u_{j_l} \in G_i$, for $1 \leq l \leq |G_i| = t$:

(a). Receive V_i and Z_i from u_i

(b). Compute $\vec{v}_{j_l}$

(c). $Y_{j_l} \leftarrow \prod_{k=1}^{n} V_i[k]^{s \cdot \vec{v}_{j_l}[k]} \bmod N^2$

(d). $S_{j_l} \leftarrow Y_{j_l}^{s \cdot \theta} \cdot Z_i[\text{Index}(\text{Cat}(u_{j_l}))]^{s \cdot (1-\theta)} \bmod N^2$

(e). Send S_{j_l} to u_i

4: User u_i:

(a). **for** $1 \leq l \leq t$ **do**:

- Receive S_{j_l} from user u_{j_l}
- $\text{SC}(u_i, u_{j_l}) \leftarrow \frac{D_{pr}(S_{j_l})}{s^3}$

user $u_{j_l} \in G_i$, the main difficulty is to compute $\text{TS}(u_i, u_{j_l})$ without revealing either $\text{Cat}(u_i)$/$\text{Tags}(u_i)$ to u_{j_l} or $\text{Cat}(u_{j_l})$/$\text{Tags}(u_{j_l})$ to u_i.

This work solves the above issue (i.e., securely computing $\text{TS}(u_i, u_{j_l}), \forall\, u_{j_l} \in G_i$) as follows. Let m be the number of deepest non-leaf category nodes in T (in order from left to right), denoted by $C_1, \ldots, C_m$. User u_i computes $\text{TS}(u_i, u'_x)$, for $1 \leq x \leq m$, where u'_x is a dummy user such that $\text{Cat}(u'_x) = C_x$. Then u_i computes the encrypted vector of scaled tag similarity scores, i.e., $Z_i[x] = E_{pk}(s^2 \cdot \text{TS}(u_i, u'_x))$, for $1 \leq x \leq m$ (the reason for scaling by s^2 instead of s will become clear later). After this, u_i sends V_i and Z_i to each user in G_i. Upon receiving the values, each user $u_{j_l} \in G_i$ performs the following operations:

- Compute the normalized friendship vector $\vec{v}_{j_l}$ using $\text{FL}(u_{j_l})$.
- Compute the encryption of (scaled) cosine similarity score between the friendship vectors of u_i and u_{j_l} as $Y_{j_l} = \prod_{k=1}^{n} V_i[k]^{s \cdot \vec{v}_{j_l}[k]} \bmod N^2$. This step is shown as step 3(c) in Algorithm 5. As mentioned earlier in the PFR_α protocol, observe that $Y_{j_l} = E_{pk}(s^2 \cdot (\vec{u}_i \bullet \vec{u}_{j_l}))$ and the scaling factor here is s^2. This is the reason for scaling $\text{TS}(u_i, u'_x)$ by the same value (i.e., s^2), for $1 \leq x \leq m$.
- The (scaled) encrypted social closeness score is computed as:

$$S_{j_l} = Y_{j_l}^{s \cdot \theta} \cdot Z_i[\text{Index}(\text{Cat}(u_{j_l}))]^{s \cdot (1-\theta)} \bmod N^2 \tag{4.5}$$

  where $\text{Index}(\text{Cat}(u_{j_l}))$ returns the index corresponding to the category node of u_{j_l}. That is, if $\text{Cat}(u_{j_l}) = C_p$, then $\text{Index}(\text{Cat}(u_{j_l})) = p$. Observe that $Z_i[\text{Index}(\text{Cat}(u_{j_l}))]$ is the encryption of (scaled) tag similarity score between u_i and u_{j_l} (i.e., $\text{TS}(u_i, u_{j_l})$). In addition, since θ (resp., $1 - \theta$) is a fractional value, one need to first multiply it by the scaling factor s before doing the exponentiation operation. After this, u_{j_l} sends S_{j_l} to u_i.

Finally, u_i gets a list of SC scores by decrypting each S_{j_l} value and dividing it by s^3 (removing the total scaling factor). Note that users in G_i with Top-K SC scores can be treated as potential candidates to be new friends of u_i; therefore, u_i can either send friend requests to them or filter them further by browsing through their profiles.

Theorem 3. *(Correctness) - For any given target user u_i and $u_{j_l} \in G_i$, the value of $\frac{D_{pr}(S_{j_l})}{s^3}$ is always equal to the social closeness score between u_i and u_{j_l}, as defined in Equation 4.1, where $1 \leq l \leq t$.*

Proof. Following from the above discussions, it is clear that $Y_{j_l} = E_{pk}(s^2 \cdot (\vec{u_i} \bullet \vec{u}_{j_l}))$ and $Z_i[\text{Index}(\text{Cat}(u_{j_l}))] = E_{pk}(s^2 \cdot \text{TS}(u_i, u_{j_l}))$. Based on these values, one can simplify Equation 4.5 as follows†:

$$\begin{aligned} S_{j_l} &= Y_{j_l}^{s \cdot \theta} \cdot Z_i[\text{Index}(\text{Cat}(u_{j_l}))]^{s \cdot (1-\theta)} \\ &= E_{pk}(s^2 \cdot (\vec{u_i} \bullet \vec{u}_{j_l}))^{s \cdot \theta} \cdot E_{pk}(s^2 \cdot \text{TS}(u_i, u_{j_l}))^{s \cdot (1-\theta)} \\ &= E_{pk}(s^3 \cdot \theta \cdot \text{NS}(u_i, u_{j_l})) \cdot E_{pk}(s^3 \cdot (1-\theta) \cdot \text{TS}(u_i, u_{j_l})) \\ &= E_{pk}(s^3 \cdot \text{SC}(u_i, u_{j_l})) \end{aligned}$$

From the above deductions, it is clear that $D_{pr}(S_{j_l}) = s^3 \cdot \text{SC}(u_i, u_{j_l})$; therefore, $\text{SC}(u_i, u_{j_l}) = \frac{D_{pr}(S_{j_l})}{s^3}$ always holds, for $1 \leq l \leq t$. □

Example 3. Refer to the sample ontology tree T as shown in Figure 4.1. Let the scaling factor s be 10 (for simplicity) and control parameter θ be 0.7. Suppose *Bob* be the target user (i.e., $u_i = Bob$) and *Charles* be the user chosen by *Bob* randomly from group G (i.e., $Charles \in G_i$ where $G_i \subset G$). As mentioned in Example 1, let their friendship vectors be $v_{Bob} = \langle 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0 \rangle$ and $v_{Charles} = \langle 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1 \rangle$ respectively. Without loss of generality, let C_3 and C_4 be the categories of *Bob* and *Charles*. That is, $\text{Cat}(Bob) = C_3$ and $\text{Cat}(Charles) = C_4$.

†To make the presentation clear, this work simply omits the $\bmod N^2$ operation in the expansion of Equation 4.5. However, this does not affect the derived results.

Note that in the PFR_β protocol the category information is computed locally; therefore, it is not revealed to other users. Based on the PFR_β protocol, various intermediate results along the process of computing social closeness score between Bob and $Charles$ are shown in Table 4.2. The final social closeness score between Bob and $Charles$ (i.e., $\text{SC}(Bob, Charles)$) is known only to Bob. □

4.5.5. Complexity Analysis of PFR_β. Next, the computation and communication costs of the PFR_β protocol are analyzed.

Computation Cost. Similar to the PFR_α protocol, the main computation cost of PFR_β comes from Stage 2. During Stage 2, the target user u_i encrypts the normalized friendship vector component-wise (following from step 2(b) of Algorithm 5) and also encrypts the tag similarity scores with each of the deepest non-leaf category nodes (from step 2(c) of Algorithm 5). This results in $O(n+m)$ encryptions. In addition, u_i has to perform t number of decryptions (from step 4(a) of Algorithm 5). As mentioned earlier, t is smaller than n in practice and also the cost of encryption and decryption are almost the same. Therefore, the computation cost of u_i is bounded by $O(n+m)$ encryptions. Furthermore, each user $u_{j_l} \in G_i$ performs n number of exponentiation operations (following from step 3(c) of Algorithm 5). Therefore, the computation cost of each u_{j_l} is bounded by $O(n)$ exponentiations, for $1 \leq l \leq t$.

Communication Cost. Unlike the PFR_α protocol, PFR_β does not need the help of network administrator for computing the users' categories; therefore, there is no communication between users and the network administrator. The only communication in the PFR_β protocol is between u_i and each user in G_i (which occurs during Stage 2 of PFR_β). During stage 2, the target user u_i sends his/her component-wise encryption of normalized friendship vector (which is of size n) and the encrypted values of m tag similarity scores to each user $u_{j_l} \in G_i$. Hence, the communication

Table 4.2: Various intermediate results during the computation of social closeness score between *Bob* and *Charles* based on the PFR_β protocol

For $s = 10$ and $\theta = 0.7$
Bob: $\vec{v}_{Bob} = \langle 0.5, 0.5, 0, 0, 0.5, 0, 0, 0, 0.5, 0, 0 \rangle$ $V_{Bob} = \langle E_{pk}(5), E_{pk}(5), E_{pk}(0), E_{pk}(0), E_{pk}(5), E_{pk}(0), E_{pk}(0), E_{pk}(0), E_{pk}(5), E_{pk}(0), E_{pk}(0) \rangle$ $Z_{Bob} = \langle E_{pk}(75), E_{pk}(75), E_{pk}(200), E_{pk}(18), E_{pk}(18), E_{pk}(18), E_{pk}(18) \rangle$ Send V_{Bob} and Z_{Bob} to *Charles*
Charles: $\vec{v}_{Charles} = \langle 0, 0.4, 0, 0, 0.4, 0.4, 0.4, 0, 0, 0, 0.4 \rangle$ $Y_{Charles} = E_{pk}(40)$ and $Z_{Bob}[\text{Index}(\text{Cat}(Charles))] = E_{pk}(18)$ $S_{Charles} = E_{pk}(334)$; send $S_{Charles}$ to *Bob*
Bob: $\text{SC}(Bob, Charles) = \frac{D_{pr}(S_{Charles})}{s^3} = 0.334$

cost between u_i and u_{j_l} is bounded by $O(z \cdot (n + m))$ in bits, where z denotes the encryption key size in bits.

4.5.6. Security Analysis of PFR_β. During the initialization step, the coordinator randomly permutes the global encrypted list of social tags before sending it to the network administrator. Due to this, the network administrator cannot trace back which social tag belongs to which user. Therefore, the revelation of global list of social tags to the network administrator can be treated as minimum information leakage. Other than this, the only communication in PFR_β is between u_i and each user u_{j_l} in G_i. Since the computation of matching scores is performed by each user locally using their respective social tags, the privacy of users social tags and their category information is preserved. In addition, the friendship vector which is sent to u_{j_l} is encrypted by u_i. Since the private key is only known to u_i, the privacy of friend lists of u_i and u_{j_l} is also preserved (following from [60, 61]). Hence, the PFR_β

protocol preserves the privacy of users' friend lists along with their social tags and category information under the assumption that the global list of social tags can be revealed to the network administrator. At the end of the PFR_β protocol, the social closeness scores (which is also the desired output) are revealed only to u_i.

4.6. OTHER SECURITY CONCERNS

This sub-section discusses additional security issues related to the disclosure of social tags and common network attacks, such as substitution, man-in-the-middle and impersonation attacks. Also, it analyzes how these attacks are related to the proposed protocols.

4.6.1. Security of Social Tags. The proposed protocols assume that social tags are part of a user's personal profile. Since social tags may reveal a user's certain interests or hobbies that the user may not want others to know, it is in the user's best interest to keep them private. Some social networks (e.g., Facebook) provide an option that allows users to set their friend lists and personal profiles as private information. Although it is not known whether this is the case for every social network, having such an option is a general trend due to increasing privacy concerns. Therefore, the main goal of the PFR_β protocol is to protect the security of users' social tags and friend lists.

In the current implementation, the PFR_β protocol may reveal the number of social tags of a user u_a to a user u_b when u_a sends his or her encrypted social tags to u_b. The size or the number of social tags hardly leaks privacy information regarding a user's profile. However, one can eliminate this problem by adding dummy social tags to u_a's encrypted social tag list to hide the actual number of social tags that u_a has. At the end, these dummy tags can be removed by the network administrator when he or she builds the social-tag or ontology tree.

4.6.2. Common Threats Related to Network Security. The proposed protocols utilize the underlying communication structure provided by the social network service provider. Therefore, as long as the underlying communication network is secure against the man-in-the-middle and impersonation attacks, the proposed protocols are safe from these attacks. On the other hand, substitution attack is possible under the proposed protocols when the adversary in consideration is malicious (instead of assuming semi-honest in the current protocols). By substitution attack, a user can substitute his or her actual input with some fake input to probe other users' private information. Since the protocols only return similarity scores, the substitution attack will not be very effective.

To further prevent substitution attack, one could employ some auditing and outlier detection protocols as follows. The system keeps an audit trail for each user. When a user with potential abnormal or malicious behavior is detected, the network administrator can run an audit check to compare the user's profile information and the actual information used by the user during the execution of the proposed protocols. This auditing process could be implemented as a secure protocol under the accountable computing framework [63, 64] so that the user's private information is not leaked during the auditing process. Developing such a secure protocol is not straightforward, and this work leaves it as part of the future research direction.

4.6.3. Dimension Reduction of the Global Friend Space. When n becomes very large, in terms of millions, the computation cost of the proposed protocol becomes very expensive. The increased cost mainly due to the size of the global user/friend space G. Some potential ways to reduce the size of the global friend space are discussed below.

Applicability of Secure Set Intersection Protocols. Since the individual friend list of a user is generally very small, one may want to use secure set intersection protocols (that do not need to represent a friend list under G) to compute the Cosine

similarity between two friend lists. However, by using the secure set intersection protocols proposed in [65, 66, 67] to compute the Cosine similarity, the intersection size between two friend lists will be disclosed. This can allow an adversary (e.g., the target user u_i) to probe other users' friend lists. To maximize security protection of friend lists, this work adopted the secure dot product protocol [68] that bypasses the direct computation of the intersection of two friend lists. Since the friend list vectors are normalized, the dot product between two normalized lists gives the Cosine similarity between the two lists. In this way, the intersection size is never disclosed.

Generating a Virtual Friend Space. When n is large and to improve computation efficiency of the proposed protocols, this work adopts universal hash functions to generate a virtual friend space that is much smaller than the actual global friend space. For example, u_i initially develops a mapping function (converting IDs in a friend list to an integer domain). This mapping function is sent to every user u_{j_l} in G_i. From the mapping function alone, u_{j_l} can map every ID or user log-in in his or her friend list to an integer value within a proper domain. Then through a universal hash function, these values can be mapped to a vector whose size is much smaller than $|G|$. Given a positive integer v, a universal hash function, denoted by $h_{a,b}$, can be derived as follows:

$$h_{a,b}(v) = (a \cdot v + b \mod p) \mod |S| \tag{4.6}$$

where p is a prime, $a \in \mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ and $b \in \mathbb{Z}_p = \{0, 1, \ldots, p-1\}$. The hash function has the property of collision resistant. The universal hash function is commonly used to map values from some larger domain to chosen smaller domain and is especially useful when the number of elements in a set is much smaller than their domain size. From the existing results [69], it has been shown that when p is close to 9,000 and the number of elements is limited by hundreds, the Cosine similarities

computed under this virtual global friend space are almost 100% accuracy. Therefore, by utilizing a universal hash function, this work claims that the proposed protocols can still be efficient and preserve maximum security without sacrificing much or any accuracy.

4.7. EXPERIMENTAL RESULTS

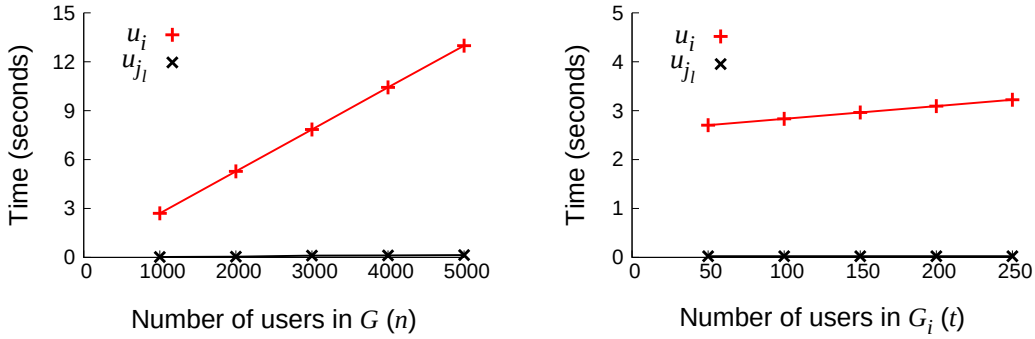
This sub-section empirically analyzes the computation costs of the proposed protocols based on various parameters. Since both proposed protocols compute the social closeness scores based on the scoring function in [13], their accuracy is exactly similar to the method in [13]. In particular to PFR_{β}, if the social closeness scores are too low (i.e., if the scores are less than a threshold value set by u_i), then u_i can repeat the process with a new subset of users (i.e., by selecting different G_i).

The proposed protocols were implemented in C, and experiments were performed on a Intel® Xeon® Six-Core™ 3.07GHz PC running Ubuntu 10.04 with 12GB of RAM. In the evaluations, Paillier cryptosystem [55] is used due to its efficiency (however, any other HPEnc system can be used to implement the proposed protocols). The Paillier key size is set to 1,024 bits (a commonly accepted key size) for all the experiments. Though the results presented here are for key size 1,024 bits, one can observe that the computation time increases by almost a factor of 6 when the size of encryption key doubles and other parameters are fixed. Furthermore, assume that on average each user $u_k \in G$ contains 100 tags (i.e., $|\text{Tag}(u_k)| = 100$). First, the computation costs of the proposed protocols are analyzed separately based on different parameters. Then, the computation costs of both protocols are compared under various parameter settings.

4.7.1. Computation Cost of PFR_α. The computation cost of PFR_α mainly depends on n (i.e., number of users in G) and t (i.e., number of users in G_i). In addition, the computation cost of u_i is different from u_{j_l} (observe that the computation cost is the same for every user in G_i). Therefore, the computation costs of u_i and u_{j_l} for different values of n and m are analyzed. Note that the computation cost of u_i and u_{j_l} are independent of m (i.e., number of non-leaf category nodes) since the computation of user category information is shifted to the network administrator in PFR_α. Hence, for the rest of this sub-section, the value of m is fixed to 20. The results are as shown in Figure 4.2.

For $t = 50$ and varying values of n, the computation cost of u_j and u_{j_l} are given in Figure 4.2(a). From Figure 4.2(a), it is clear that the computation of u_i is significantly high (due to expensive encryption operations) compared to u_{j_l} (due to less expensive homomorphic addition and multiplication operations). For example, when $n = 1000$, the computation cost of u_i and u_{j_l} are 2.703 and 0.024 seconds respectively. As expected, the computation cost of u_i and u_{j_l} is almost doubled whenever n is increased by a factor of two. For instance, as shown in Figure 4.2(a), the computation cost of u_i is changed from 2.703 to 5.277 seconds when n is changed from 1000 to 2000. However, for u_{j_l}, the increase in the computation cost is very small (due to less expensive homomorphic addition and multiplication operations) when n increases.

Similarly, for $n = 1000$ and varying values of t, the computation cost of u_j and u_{j_l} are as shown in Figure 4.2(b). Since the computation cost of u_{j_l} is independent of t, it remains constant (0.024 seconds) for varying values of t. On the other hand, the number of decryption operations performed by u_i depends on t. As shown in Figure 4.2(b), the computation cost of u_i grows linearly with t. For example, the computation cost of u_i varies from 2.703 to 3.223 seconds when t is changed from 50 to 250.



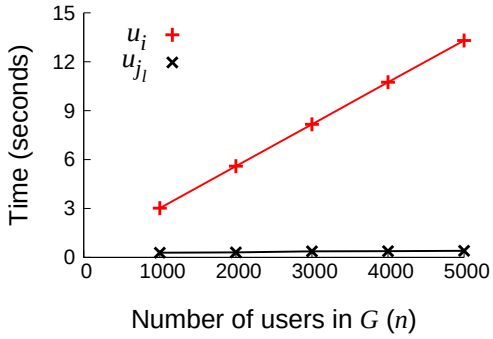
(a) For $t = 50$ and $m = 20$ (b) For $n = 1000$ and $m = 20$

Figure 4.2: Computation costs of PFR_α for varying n and t with $m = 20$

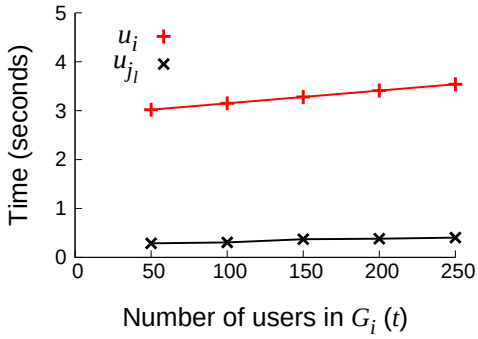
4.7.2. Computation Cost of PFR_β. Unlike PFR_α, the computation cost of PFR_β also depends on m in addition to the parameters n and t. Therefore, the run time of u_i and u_{j_l} for varying values of n, t, and m are computed. The results are as shown in Figure 4.3.

First, the values of t and m are fixed to 50 and 20, respectively. The computation cost of u_i and u_{j_l} for varying values of n are shown in Figure 4.3(a). It is clear that, following from Figure 4.3(a), the computation cost of both u_i and u_{j_l} grows linearly with n. For example, when n is changed from 1000 to 5000, the computation cost of u_i varies from 3.02 to 13.304 seconds respectively. A similar trend can be observed for u_{j_l}.

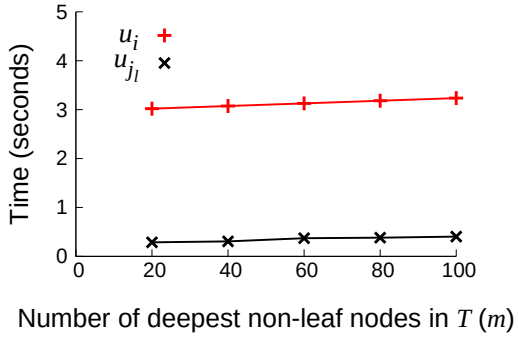
Next, when $n = 1000$ and $m = 20$, the run time of u_i and u_{j_l} for varying t are computed. As shown in Figure 4.3(b), the computation cost of u_i varies from 3.02 to 3.54 seconds when t is changed from 50 to 250. Whereas, the computation cost of u_{j_l} varies from 0.287 to 0.404 seconds when t is varied from 50 to 250. In a similar way, the run time of u_i and u_{j_l} for varying values of m are computed when $n = 1000$ and $t = 50$. As shown in Figure 4.3(b), observe a linear growth in the computation time of both u_i and u_{j_l} with varying m.



(a) For $t = 50$ and $m = 20$



(b) For $n = 1000$ and $m = 20$

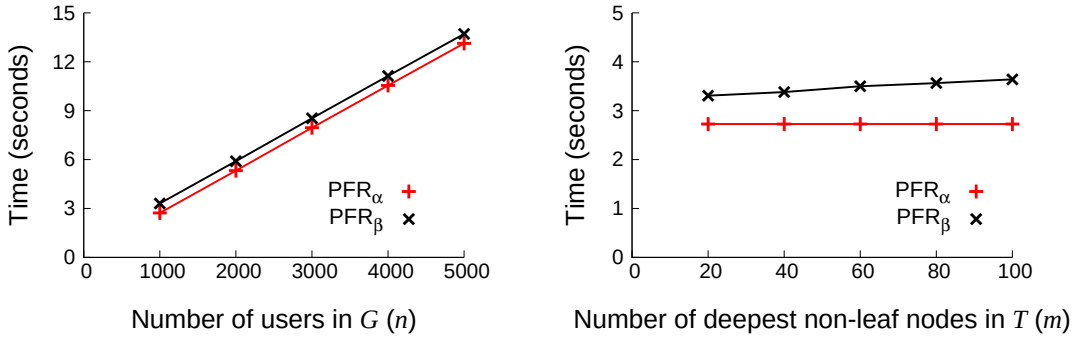


(c) For $n = 1000$ and $t = 50$

Figure 4.3: Computation complexity of PFR_β for varying n, t, and m

Based on the above results, it is clear that the computation cost of u_i and u_{j_l} in both PFR_α and PFR_β does not increase much for varying t. This further justifies the complexity analysis given in Sections 4.5.2 and 4.5.5.

4.7.3. Comparison: PFR_α Vs. PFR_β. This sub-section compares the total run time of PFR_α with PFR_β (excluding the cost of the network administrator) as shown in Figure 4.4. First, the values of t and m are fixed to 50 and 20, respectively. As shown in Figure 4.4(a), the total run time of both protocols grows linearly with increasing n. Note that since $|\text{Tag}(u_k)|$ is fixed to 100 for each user $u_k \in G$, the total run time of PFR_β is not significantly more compared to that of PFR_α for any given n. For example, when $n = 3000$, the total run time of PFR_α and PFR_β are 7.953 and 8.533 seconds, respectively.



(a) For $t = 50$ and $m = 20$

(b) For $n = 1000$ and $t = 50$

Figure 4.4: PFR_α Vs. PFR_β

Given $n = 1000$ and $t = 50$, the total run time of both proposed protocols for varying values of m are as shown in Figure 4.4(b). As expected, the run time of PFR_α (which is 2.727 seconds) remains to be constant for increasing values of m since the computation cost of both u_i and u_{j_l} in the PFR_α protocol are independent of m. However, the run time of PFR_β varies from 3.307 to 3.64 seconds when m is increased from 20 to 100 since the computation cost of u_i in the PFR_β protocol depends on m.

4.8. CONCLUSION

In the recent years, friend recommendation application has gained significant importance since it helps for the development of entire network structure which is essential for the survival of any online social network. In addition, it also helps users to meet new friends based on their dynamic interests and to expand their social connections. However, the existing friend recommendation techniques do not take users' privacy into consideration; therefore, they are not applicable in a privacy-preserving environment. This work proposed two private friend recommendation algorithms for users in a group G by leveraging both social tags and network topology.

For a given target user u_i, the proposed protocols compute the social closeness scores between u_i and each user in the subset $G_i \subset G$ in a privacy-preserving manner by utilizing an ontology tree T constructed by the domain expert such as the network administrator.

In the first protocol (referred to as PFR_α), the target user u_i can choose the users in G_i (excluding u_i and friends of u_i) based on his/her interest. Also, PFR_α is more efficient than the second protocol (referred to as PFR_β), however, this efficiency gain comes at the expense of releasing users social tags to the network administrator. Whereas the PFR_β protocol facilitates user u_i to compute the social closeness scores with some random users in G (i.e., G_i is chosen randomly after excluding u_i and friends of u_i from G). Nevertheless, PFR_β protects the privacy of users' friend lists, social tags, and category information.

The proposed protocols in this work assume that the network administrator builds the ontology tree T based on the domain knowledge of a particular group. However, extending it to multiple groups may not seem to be feasible at this point of time and this issue will be addressed in the future work. Though the second protocol is more secure, the amount of computation involved for user u_i is high when the number of deepest non-leaf (category) nodes in T is large. In addition, this work used social tags as the content of user information. However, computing the similarity between user profiles based on other details such as education, hobbies, and employment in a privacy-preserving manner can still be achieved using the secure computation of cosine similarity. Therefore, investigation of the above issues can be regarded as an interesting direction for future work.

5. PRIVACY-PRESERVING FRIEND RECOMMENDATIONS USING COMMON NEIGHBORS METHOD

Given the snapshot of an OSN, the social closeness between two users is termed as proximity. The proximity measures can be divided into two groups. The first group includes measures based on node neighborhoods, such as common neighbors, Jaccard Coefficient, Adamic/Adar, and preferential attachment. Whereas, the second group consists of measures based on ensemble of all paths, such as Katz, Hitting time, Page Rank and SimRank. It was shown that the common neighbors method acts as a good proximity measure to capture the similarity among the nodes of a social network [15, 16]. Therefore, to be more concrete, this work uses the common neighbors as the similarity measure to compute the social closeness between two users.

More specifically, this work proposes two novel methods for solving the friend recommendation in a privacy-preserving manner. What is private to a given user is subjective in nature. In particular to the friend recommendation problem, the friendship between any two users can be treated as sensitive/private information. This assumption is realistic and being supported by many on-line social networks (e.g., Facebook) where users are allowed to hide their friend lists. Most often, when people maintain friendship with trusted ones, there is much information flowing from one to another. Thus, revealing this sensitive information (i.e., friendship) poses a great threat to user privacy through social engineering attacks [24, 25].

Additionally, this work assumes that the social network operators or administrators are not allowed to know the users' friend lists or other private personal profile information. Note that this is a commonly made assumption in the related problem domains [57, 70, 71, 72, 73, 74, 75, 76] and such a social network has several significant benefits. For example, when a user's private personal profile is not disclosed to the social network, the network would likely not face any obligations about leaking

its users' private information when the network were hacked. Therefore, the social network has the incentive not to know the user's private information, like friend list. On the other hand, the user has more incentive to use the social network since the user has complete control over his or her private information. Due to increasing privacy concerns, such a private social network will become more common, and it can attract non-traditional users. In addition, since the social network provider does not know the network structure and users' private profile, this may allow private organizations or government agencies to outsource their internal communication means to such networks. This work claims that there will be a market for such a privacy social network. Hence, by assuming users' friend lists are private, this work proposes a set of solutions by utilizing the common neighbors method as the underlying similarity metric.

5.1. PROBLEM DEFINITION

Consider a social network where the friendship of two users is treated as private, and the other users should not know whether the two users are friends. In this social network, users can still create or share contents among themselves, but the friend list of a user (considered private) should be accessible only by the user himself/herself. The challenge is how to perform friend recommendation in such a social network without disclosing the friendship information between any two users to the other users. This problem is referred to as privacy-preserving friend recommendation (PPFR), and the goal of this section is to develop PPFR protocols based on the common neighbors method [15]. Let A be the target user (who wish to make new friends) in a social network and $Fr(A)$ denote the friend list of A. Without loss of generality, assume $Fr(A) = \langle B_1, \ldots, B_m \rangle$, where B_i is a friend of A, for $1 \leq i \leq m$.

More formally, the PPFR protocol can be defined as follows:

$$\text{PPFR}(A, Fr(A), Fr(B_1), \ldots, Fr(B_m), t) \rightarrow S \tag{5.1}$$

where t denotes a threshold value (more details are given in the later part of this section) and S denotes the list of recommended friends whose common neighbors scores with A are greater than or equal to t. The following properties hold for a PPFR protocol:

(a). **Privacy-Preserving**

- $Fr(A)$ (resp. $Fr(B_i)$, for $1 \leq i \leq m$) is never disclosed to users other than A (resp. B_i) in a social network.
- Similarly, $\forall\ C_{ij} \ \in \ Fr(B_i)$, $Fr(C_{ij})$ is never disclosed to users other than C_{ij}.
- Friend lists of all users including A are never disclosed to the social network administrator T.
- At the end of PPFR, S is only known to A.

(b). **Correctness**

- $S \subseteq \bigcup_{i=1}^{m} Fr(B_i)$
- $\forall\ C \in S \quad \Rightarrow \quad \Phi(A, C) \geq t$

where C is a recommended new friend to user A and $\Phi(A, C)$ denotes the common neighbors score between A and C. In order to recommend C as a new friend to A, $\Phi(A, C)$ should be greater than or equal to t.

The proposed PPFR protocols provide a means to recommend friends to A based on the common neighbors scores in a privacy-preserving manner. Note that only the

users with scores $\geq t$ are recommended as friends to A. However, A may not wish to make friendship just simply based on the recommendations from the PPFR protocols. That is, A might want to know more about the recommended friends before sending the friend requests. Nevertheless, after the recommendations, A can visit the web pages corresponding to the recommended friends and send friend requests to only those users who are of particular interest to A.

For example, consider the case where Bob and Charles are recommended as new friends to Alice using the PPFR protocols. Without loss of generality, let us assume that Charles was in the same high school as Alice and Bob has no relationship with Alice (except that they share one or more common friends). Once Alice receives the recommendations, she may find that Charles is her old schoolmate from the webpage of Charles. Then, Alice can simply send a friend request only to Charles (whom she might trust) and may ignore Bob (if Alice does not want to establish friendship with unknown users).

5.2. MAIN CONTRIBUTIONS

This work presents two novel PPFR protocols that satisfy the well-known security definitions in the literature of secure multiparty computation (SMC) [50, 77]. The first protocol adopts an additive homomorphic encryption scheme [55] and provides a very strong security guarantee. It also utilizes a universal hash function for improving the efficiency. This efficiency comes at the expense of degraded accuracy due to the involved hash collisions. Whereas, the second method utilizes the concept of protecting the source privacy [78] through randomizing the message passing path and recommends friends accurately. The main contributions of this work are summarized as follows:

- **Security -** Both of the proposed protocols preserve the privacy of each user (i.e., his/her friend list). Nevertheless, both protocols leak different additional information thereby providing different security guarantees. The first protocol leaks the common neighbors scores to a third party (denoted by T, such as the network administrator). However, due to hashing and random permutation, T cannot identify the source of this scores. The second protocol reveals the common neighbors scores which are $\geq t$ to user A. Since the protocol guarantees the source privacy, A cannot determine the sources corresponding to the scores. A detailed security analysis of both protocols is provided in the later part of this section.

- **Accuracy & Efficiency -** The first protocol uses a universal hash function, so it is expected to produce false positives and false negatives due to hash collisions. Also, its efficiency depends on the parameters of hash function and the size of $Fr(A)$. On the other hand, the second protocol recommends friends accurately and its efficiency depends mainly on the size of both $Fr(A)$ and friend lists of A's friends.

- **Flexibility -** The proposed protocols act as a trade-off among security, accuracy and efficiency; therefore, a domain expert can choose between these two protocols depending on the requirements of a specific application.

5.3. RELATED WORK

This sub-section first reviews upon the existing work related to friend recommendations in OSNs. Then it discusses the private social networks and the existing PPFR algorithms. Also, it summarizes the literature work on secure multiparty computation along with the security definition adopted in this paper.

5.3.1. Existing Friend Recommendation Algorithms in OSNs. Social network analyses have been utilized for various business applications [6, 27], such as predicting the future [28] and developing recommender systems [29, 30, 31, 32]. With growing interest of expanding a person's social circle, friend recommendation has become an important service in many OSNs. Along this direction, researchers from both academia and industry have published much work. In particular, Chen et al. [9] evaluated four recommender algorithms, which utilize social network structure and/or content similarity, in an IBM enterprise social networking site Beehive through personalized surveys. Their analysis showed that algorithms based on social network information produce better-received recommendations. A novel user calibration procedure was proposed by Silva et al. [11] based on a genetic algorithm to optimize the three indices derived from the structural properties of social networks. Xie [12] designed a general friend recommendation framework to recommend friends based on the common interests by characterizing user interests in two dimensions - context (e.g., location and time) and content.

By treating friend recommendation process as a filtering problem, Naruchitparames et al. [10] developed a two-step approach to provide quality friend recommendations by combining cognitive theory with a Pareto-optimal genetic algorithm. Gou et al. [13] developed a visualization tool (named as SFViz) that allows users to explore for a potential friend with an interest context in social networks. Their method considers both semantic structure in social tags and topological structures in social networks to recommend new friends. The correlation between social and topical features in three popular OSNs: Flickr, Last.fm, and aNobii has been studied by Aiello et al. [33] to analyze friendship prediction. Their results showed that social networks constructed solely from topical similarity captured the actual friendship accurately. Nevertheless, Facebook uses the "People You May Know" feature to recommend friends based on the simple "friend-of-a-friend" approach [34].

5.3.2. Private Social Networks. In current online social networks (OSNs), such as Facebook and Google+, users do not have full control of their own data. Though the OSN providers facilitate users with various privacy options, this will not guarantee the privacy of user's data since the data are stored on the server of an OSN provider (assuming the data are not encrypted). In addition, with the past history of data leaks and privacy controversies [79, 80], users have many trust issues with OSN providers. To address various security concerns in OSNs, researchers have been working to develop decentralized architectures for OSNs where social networking service is provided by a federation of nodes (i.e., peer-to-peer networks). Along this direction, much work has been published such as [81, 82, 83, 84, 85, 86].

Most recently, Nilizadeh et al. [87] proposed a decentralized architecture (referred to as Cachet) to efficiently support users with the central functionality of OSNs while providing security and privacy guarantees. Cachet uses a combination of techniques, namely distributed hash tables and attribute based encryption [88], to protect confidentiality, integrity, availability of user content as well as the privacy of user relationships. In particular, they developed a gossip-based social caching algorithm to speedup the process of loading the data in newsfeed application. However, their scheme is entirely different from the work presented in this section. First, their decentralized architecture involves no network provider and the communication happens directly between users. In this work, user's data are encrypted and stored on the server of an OSN. Therefore, the problem setting in this paper is orthogonal to their model. Second, the algorithm proposed in [87] is to support newsfeed application (under peer-to-peer network architecture) whereas this paper focuses on friend recommendation application. Third, the peer-to-peer based social network system incurs heavy computation as well as communication costs on the users. Therefore, this work focuses on conventional OSN framework where the data resides on the server of an OSN.

The decentralized architecture in OSNs is a more restricted model since it avoids the use of an OSN provider altogether. Also, it was claimed in [76] that decentralization is an insufficient approach since it leaves the user with an enviable dilemma: either sacrifice availability, reliability and convenience by storing data either on his/her machine or entrust his/her data to one of the several providers that he/she probably does not know or trust any much more than he/she would with a centralized provider. Therefore, this work explicitly assumes that user's data are encrypted at first place, for privacy and security reasons, and then sent to an OSN. Under such an architecture, where user's encrypted data are stored on the OSN's server, various schemes, such as [57, 70, 71, 72, 73, 74, 75], have been proposed to protect users' privacy through cryptography. Nevertheless, even after proper encryption of user's data, centralized provider cannot be trusted for various reasons. For example, a malicious provider can show different users divergent views of the system state. This behavior is referred to as server equivocation [76].

Along this direction, the authors in [76] developed a novel framework for social network applications, referred to as Frientegrity, that can be realized with an untrusted service provider. In Frientegrity, the service provider sees only the encrypted data and cannot deviate from correct execution without being detected. Therefore, their system preserves data confidentiality and integrity. However, as mentioned in [76], Frientegrity reveals social relations to the service provider whereas in this work social relations (i.e., friend lists) are protected from the service provider. Also, in their method, the target user A discovers the new friends by searching through the access control lists of his/her friends for potential FoF (i.e., friend-of-a-friend) that he/she might want to become friend with. However, such a process is not allowed from user's privacy perspective since this reveals the friend lists of A's friends to A. On the other hand, in this work, friend list of a user is never revealed to other users.

This work assumes that users as well as the OSN provider are semi-honest. However, in the case of malicious model, the proposed protocols can be combined with the techniques from [76] in order to mitigate the server equivocation problem.

5.3.3. Existing PPFR Algorithms in OSNs. Due to various privacy issues [17, 18, 19, 20, 21, 22, 23], many users keep their friend lists private. Only recently, researchers have focused on developing accurate and efficient PPFR methods. Along this direction, Dong et al. [35] proposed a method to securely compute and verify social proximity between two users using cosine similarity in mobile social networks. This work differs from theirs in two aspects. First, the problem setting in this section is entirely different from theirs. More specifically, this work assumes users' friend lists are kept as secret; thus, who are involved in the computation is not revealed before hand. Whereas, in their approach, participating users are aware of one another, and only their social closeness is securely computed and verified to determine the new friendship. Secondly, the proposed protocols in this work are purely based on the user IDs and does not need any pre-computation from a server. On the other hand, their approach assumes that the social coordinates (which may change often due to the mobility of users) for individual users are pre-computed by a trusted central server which is a violation of user privacy.

Machanavajjhala et al. [36] estimated the trade-offs between accuracy and privacy of private friend recommendations using differential privacy [37, 54]. In their work, the authors used the existing differentially private algorithms as underlying sub-routines and assumed the existence of PPFR protocols based on these sub-routines. Also, according to their claims, if privacy is to be preserved when using the common neighbors utility function, only users with $\Omega(\log n)$ friends can hope to receive accurate recommendations, where n is the number of users in the graph. In this work, the privacy guarantees are based on an entirely different security model, namely the semi-honest security definitions from the field of secure multiparty computation

(SMC) [38, 39, 50]. Under the SMC model, this work develops accurate PPFR protocols. In particular, the second proposed protocol recommends friends accurately irrespective of the size of users' friend list and simultaneously preserves the privacy of each user.

Recently, Samanthula et al. [89] proposed a new private friend recommendation algorithm using a specific scoring function [14, 53] that takes the social network structure as well as the number of real message interactions among social network users into consideration. For a given target user A, their method computes the recommendation scores of all potential users who reside within a radius of r (≥ 2) in the corresponding candidate network of A. When $r = 2$, the snapshot of the social network in their approach is the same as the one considered in this work. Nevertheless, the scoring function adopted in this work (i.e., common neighbors method) is entirely different from theirs, and the protocols proposed in this section are more secure in the perspective of protecting friendship information.

5.3.4. Secure Set Operations. At the first glance, secure set operation protocols [66, 67, 90] related to intersection and union may be adopted to solve the proposed friend recommendation problem. There are two main challenges to use these secure set operations protocols: one is related to computation efficiency and the other is related to security. For example, let illustrate the challenges using the protocols proposed in [67] since these protocols seem most suitable to solve the proposed friend recommendation problem.

First of all, to generate encrypted polynomial representation of friend lists of A's friends is computationally expensive because the polynomials $f_1, \ldots, f_m$ from all users $B_1, \ldots, B_m$ need to be multiplied together to generate $f = \Pi_{i=1}^{m} f_i$ and f needs to be computed based on the encrypted coefficients of $f_1, \ldots, f_m$. Since the degree or the number of coefficients of f_i discloses the size of the friend list of B_i, one need to

hide the actual degree. Let make the matter simple and assume $f_1, \ldots, f_m$ have the same degree denoted by d. Therefore, the degree of f is bounded by $m \cdot d$.

According to [67], the computation of f needs to be done sequentially. That is, B_1 sends the encrypted f_1, denoted $E(f_1)$ to A (serving as a router), where $E(f_1) = \{E(f_1[d]), \ldots, E(f_1[0])\}$ and $f_1[j]$ is a coefficient for $0 \leq j \leq d$. Then A sends $E(f_1)$ to B_2, and B_2 returns $E(f_1 \cdot f_2)$ to A. B_2 needs to perform d^2 exponentiations to produce $E(f_1 \cdot f_2)$. A repeats the above process with each remaining user. However, the computation increases from B_i to B_{i+1}. Specifically, the computation cost of the i^{th} user B_i is bounded by $i \cdot d^2$ exponentiations. For B_m, the cost is $m \cdot d^2$. As a result, even when d and m are small (e.g., $m = 50$ and $d = 200$), the computation costs for some users are much higher than the proposed protocol (detailed complexity of the proposed protocols is given in the later part of this section).

In addition, more computation is needed to actually identify the element in the union above the threshold. In particular, it needs to call another secure protocol "IsEq" to verify if two ciphertexts correspond to the same plaintext. The "IsEq" protocol is the same as the "testRecord" protocol given in [91]. The "testRecord" protocol is not straightforward to implement, and it needs to be called $m \cdot d$ times. This further reduces the computation efficiency comparing to the proposed protocols. Furthermore, it takes m sequential rounds for A to compute $E(f)$, so the throughput in the proposed protocol is much higher.

Moreover, since the polynomial f needs to be evaluated by each B_i based on his or her private dataset. This evaluation will not produce the correct result if A randomizes $E(f)$ in order to hide the actual user IDs. On the other hand, if A does not randomize $E(f)$, B_i will know who (among his or her current friend list) will be recommended as new friends for A. It is not clear how to prevent this information leakage if one adopts the aforementioned secure set operation protocols.

5.4. PRELIMINARIES

This sub-section presents some concepts and properties related to universal hash functions, additive homomorphic encryption schemes, and common neighbors methods which are used throughout the section. Some of the common notations used in this section are highlighted in Table 5.1.

5.4.1. Universal Hash Function. Consider a set of integers $L = \{0, 1, \ldots, l-1\}$. The goal is to map the integers from domain L to a smaller domain $V = \{0, 1, \ldots, s-1\}$ (where $s < l$) with minimum number of collisions. This can be achieved by a universal hash function [92]. Given a positive integer $x \in L$, a universal hash function $h_{a,b}$ is defined as follows:

$$h_{a,b}(x) = (a \cdot x + b \bmod p) \bmod s \tag{5.2}$$

where p is a prime $\geq l$, a and b are randomly chosen from $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ and $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ respectively. The universal hash function has the property of collision resistant. This is because, $\forall x, y \in L$, $h(x) - h(y) \bmod v$ is uniformly distributed in V. That is the chance of collision between x and y is $\frac{1}{s}$. (Note that there is no perfect hash function without collisions.)

5.4.2. Additive Homomorphic Probabilistic Encryption. The proposed first protocol uses an additive homomorphic encryption scheme (denoted as HEnc) that is probabilistic in nature. Let E and D be the encryption and decryption functions of an HEnc system with pk and pr as their respective public and private keys. Suppose N^* is the RSA modulus or the group size, and it is a part of the public key pk. Given two plaintexts $x_1, x_2 \in \mathbb{Z}_N$, an HEnc system has the following properties:

- The encryption function is additive homomorphic: $E_{pk}(x_1) \cdot E_{pk}(x_2) = E_{pk}(x_1 + x_2)$.

*The product of two large prime numbers of similar bit-length.

Table 5.1: Common notations used in the PPFR protocols

HEnc	An additive homomorphic probabilistic encryption scheme
PPFR	Privacy-preserving friend recommendation
SMC	Secure multiparty computation
$\langle E, D\rangle$	A pair of encryption and decryption functions in an HEnc System
$\langle pk, pr\rangle$	Public and private key pair corresponding to $\langle E, D\rangle$
$h_{a,b}(x)$	Hash value of an integer x, and $h_{a,b}$ is an universal hash function
s	The domain size of the hash function $h_{a,b}$
$Fr(A)$	The friend list of user A
B or B_i	One friend of A
C or C_{ij}	One friend of B or B_i, or any user in a social network
t	A threshold value for recommendation condition

- Given a constant $c \in \mathbb{Z}_N$ and $E_{pk}(x_1)$: $E_{pk}(x_1)^c = E_{pk}(c \cdot x_1)$.
- The encryption function has semantic security as defined in [56]. Briefly, a set of ciphertexts do not provide additional information about the plaintext to an adversary.

There are many HEnc systems available in the literature. However, this work uses the Paillier's encryption scheme [55] due to its efficiency.

5.4.3. The Common Neighbors Method. Let A and C be two users in a social network such that $C \notin Fr(A)$. Then, the common neighbors score (denoted as Φ) for A and C is defined as the number of friends/neighbors that A and C have in common [93]. More formally,

$$\Phi(A, C) = |Fr(A) \cap Fr(C)| \tag{5.3}$$

Note that the common neighbors score is computed between two-hop neighboring nodes. Therefore, in order to recommend new friends for user A, consider the users who are two hops away from A as the possible candidates.

Example 3. *Consider the sample two-hop snapshot of the social network for Alex as shown in Figure 5.1. The figure shows all the users who are at most two hops away from Alex. The goal is to recommend new friends to Alex. Let us assume that* $t = 2$. *From Figure 5.1, it is clear that the set of potential candidates are* $\langle Baker, Bob, Evans, Martin, Wilson \rangle$. *The friend lists and the common neighbors scores for Alex and each one of the potential candidates are as shown in Table 5.2. Since* $\Phi(Alex, Baker) = 2$ *and* $\Phi(Alex, Martin) = 2$, *Baker and Martin are recommended as new friends to Alex.*

5.5. PROPOSED PRIVACY-PRESERVING FRIEND RECOMMENDATION PROTOCOLS

To solve the PPFR problem, this work considers the following practical assumptions supported in many online social networks:

1. **Friendship is symmetric**: If A is in B_i's friend list, then B_i must be in A's friend list.
2. **Known Participation**: A user participation in the social network is public information. E.g., a person can browse existing Facebook user IDs.
3. **Recommendation Condition**: Consider the case of recommending new friends to A. These new friends are chosen from the candidate users that share at least t friends with A. Here, the candidate users are all the friends of A's friends. In general, t can be a static value decided by either the network administrator or A, and it can be changed occasionally. The observation is $1 \leq t \leq |Fr(A)|$.

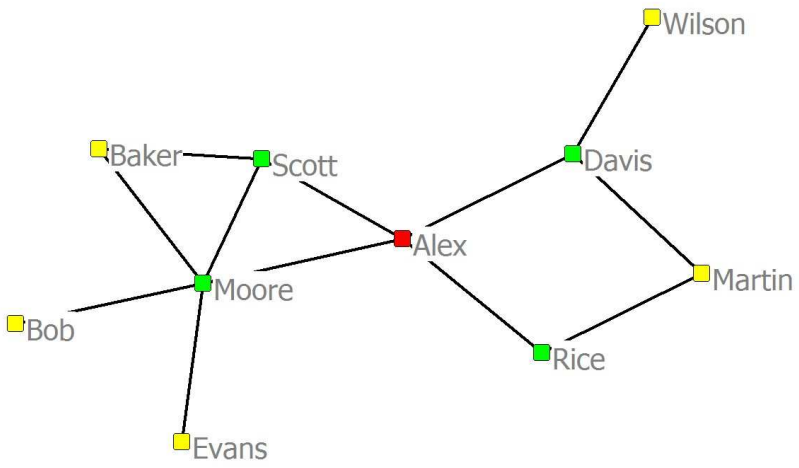


Figure 5.1: Sample snapshot (two-hop) of social network for *Alex*

On one hand, a small value of t may result in more number of recommendations. On the other hand, while increasing it, may result in lesser number of recommendations. In this section, assume that t (which is randomly chosen by A from $[1, |Fr(A)|]$) is public.

Since the proposed protocols are distributed, the messages communicated among participating entities need to be authenticated. There exist many user and message authentication protocols, so this work does not address these issues when presenting and analyzing the proposed protocols. In addition, this work assumes that the participating entities do not collude.

5.5.1. The PPFR$_h$ Protocol. The PPFR$_h$ protocol is based on an additive homomorphic encryption scheme as explained earlier. The basic idea is to encrypt the ID of each friend of A's friend independently (under a common candidate space) and perform a homomorphic addition by A on the encrypted data. Since the global user space is public in a social network, one could use this global space to represent the candidate space without disclosing any particular candidate's ID to A. However,

Table 5.2: Friend lists and the computation of common neighbors scores for *Alex* based on Figure 5.1

Friend Lists		
$Fr(Alex)$	$=$	$\langle Scott, Moore, Rice, Davis\rangle$

candidates $= \langle Baker, Bob, Evans, Martin, Wilson\rangle$

$Fr(Baker)$	$=$	$\langle Scott, Moore\rangle$
$Fr(Bob)$	$=$	$\langle Moore\rangle$
$Fr(Evans)$	$=$	$\langle Moore\rangle$
$Fr(Martin)$	$=$	$\langle Rice, Davis\rangle$
$Fr(Wilson)$	$=$	$\langle Davis\rangle$

Common neighbors scores for *Alex*		
$Fr(Alex) \cap Fr(Baker)$	$=$	$\langle Scott, Moore\rangle$
$Fr(Alex) \cap Fr(Bob)$	$=$	$\langle Moore\rangle$
$Fr(Alex) \cap Fr(Evans)$	$=$	$\langle Moore\rangle$
$Fr(Alex) \cap Fr(Martin)$	$=$	$\langle Rice, Davis\rangle$
$Fr(Alex) \cap Fr(Wilson)$	$=$	$\langle Davis\rangle$
$\Phi(Alex, Baker) = \Phi(Alex, Martin) = 2$		
$\Phi(Alex, Bob) = \Phi(Alex, Evans) = 1$		
$\Phi(Alex, Wilson) = 1$		

as the size of the social network is usually large (e.g., in millions), generating a global user space is often impractical.

In general, the size of the potential candidate space (i.e., $\sum_{i=1}^{m} |Fr(B_i)|$) is much smaller than the size of the global user space. Additionally, since users' friend lists are private, neither A nor B_i knows the potential candidate space before hand. A quick fix is for B_i to use a universal hash function that can hash the IDs of his/her friends into integers less than s (a predefined system parameter). This creates a pseudo and oblivious candidate space. Since the universal hash function maps integers from one integer domain into another integer domain, there is a need to have a mechanism to convert each user ID which may contain some characters into a unique integer. A simple and straightforward approach is to initially convert each character

in the ID (if there are any) into its ASCII value (a three-digit integer). In addition, a single digit d ($0 \leq d \leq 9$) can be mapped to $128 + d$. Then, any given string can be mapped to a unique integer by appending these three-digit values together. E.g., ID "$Bob52$" is converted as follows:

$$\begin{aligned} int(Bob52) &= \text{ASCII}(B)||\text{ASCII}(o)||\text{ASCII}(b)||133||130 \\ &= 066||111||098||133||130 \\ &= 066111098133130 \end{aligned}$$

where $||$ denotes concatenation. This conversion method satisfies the condition that each user ID is mapped into a unique integer, and this integer can be easily mapped back to its corresponding user ID. The collision caused by the hash function is the price to pay for the accuracy in recommendations. Hereafter, A is used to simply represent the converted integer of ID(A) for convenience. Also, $Fr(A)$ gives the list of converted user IDs of A's friends.

Based on the above description, A chooses a prime number p, such that the converted user IDs are in the range of $[0, \ldots, p-1]$. Then A chooses the values of a and b randomly from Z_p^* and Z_p respectively. In addition, A chooses the domain size of the hash function $h_{a,b}$ and sets it to s. Also assume that there exists a party T (e.g., the network administrator) in the network which generates the key pair (pk, pr) using the Paillier's encryption scheme [55]. T sends the public key pk to A. In this protocol, T is only responsible for certain intermediate computations. These computations do not reveal the user IDs to T (due to involved randomization and permutation) and consequently preserve user privacy.

The main idea of the PPFR$_h$ protocol is as follows. Initially, each B_i hashes the user IDs in his/her friend list (omitting A's ID) and generates a two-column matrix M_i, where the first column contains user IDs and the second column contains either 0s

Algorithm 6 PPFR$_h$

Require: Friend list of each user is treated as private (Note: the public key pk is known to A and T, whereas the private key pr is known only to T)

1: A:

(a). Randomly choose a, b from $\mathbb{Z}_p^*$ and $\mathbb{Z}_p$, and pick s

(b). Send $h_{a,b}$ (i.e., a, b, p, s) and pk to each B_i $(1 \leq i \leq m)$

2: B_i $(1 \leq i \leq m)$:

(a). Receive $h_{a,b}$ and pk from A

(b). Compute matrix M_i using $h_{a,b}$ and $Fr(B_i)$

(c). Compute $M_i'[j][k] \leftarrow E_{pk}(M_i[j][k])$, for $1 \leq j \leq s$ and $1 \leq k \leq 2$

(d). Send M_i' to A

3: A:

(a). Receive M_i' from B_i, for $1 \leq i \leq m$

(b). Compute $Z[j][2] \leftarrow \Pi_{i=1}^{m} M_i'[j][2]$, for $1 \leq j \leq s$

(c). Compute $Z[j][1] \leftarrow Z[j][2]^{r_j} \cdot \Pi_{i=1}^{m} M_i'[j][1]$, where r_j is randomly chosen from $\mathbb{Z}_N^*$, for $1 \leq j \leq s$

(d). Compute $\hat{Z} \leftarrow \pi(Z)$ and send $\hat{Z}$ to T (note that function π is known only to A)

4: T:

(a). Receive $\hat{Z}$ from A

(b). Compute $freq_j \leftarrow D_{pr}(\hat{Z}[j][2])$, for $1 \leq j \leq s$

(c). If $freq_j \geq t$, compute $\beta_j \leftarrow \frac{D_{pr}(\hat{Z}[j][1])}{freq_j}$. Else, set β_j to 0, for $1 \leq j \leq s$

(d). Send β to A

5: A:

(a). Receive β from T

(b). $F \leftarrow \pi^{-1}(\beta)$

(c). $S = \{F_j - r_j \bmod N \mid F_j \in F \ \wedge \ F_j \neq 0, \text{for } 1 \leq j \leq s\}$

or 1s. Given the j^{th} row in M_i, if the value in the second entry is 1, the corresponding first column entry contains an actual user ID, and j equals to the hashed value of the

user ID. After successfully generating M_i, B_i encrypts M_i component-wise and sends it to the target user A.

Upon receiving all encrypted matrices, A computes the encrypted two-column frequency matrix using homomorphic additions, where the first column contains encrypted user IDs and the second column contains the encrypted frequency of the corresponding user ID in the friend lists of A's friends. After this, A randomizes the user IDs, permutes the frequency matrix row-wise and sends the resulting encrypted matrix to T. Upon receiving the matrix, T decrypts the second entry of each row to get the frequency. If the second entry is greater than or equal to t, he/she decrypts the corresponding first entry. Else, because the frequency is less than t, there is no need to recommend the corresponding user. Therefore, T sets the corresponding first entry to zero. Note that the first entries are randomized by A, so the decrypted values will result in random values. Then T sends the updated first entries to A. Finally, A performs the inverse permutation over the entries received from T and removes the randomization for all non-zero entries to get the actual set of recommended user IDs as new friends.

The main steps involved in the PPFR$_h$ protocol are shown in Algorithm 6. To start with, whenever A wants to make new friends, he/she shares the parameters of $h_{a,b}$ (i.e., a, b, p, s) and pk with only his/her friends. Then each friend B_i of A generates a matrix M_i of size $s \times 2$ whose values are initialized to zero and updated according to $Fr(B_i)$ as follows:

$$\begin{aligned} M_i[h_{a,b}(Fr(B_i)[j])][1] &= Fr(B_i)[j] \\ M_i[h_{a,b}(Fr(B_i)[j])][2] &= 1 \end{aligned}$$

where $Fr(B_i)[j]$ is the j^{th} user in $Fr(B_i)$, for $1 \leq i \leq m$ and $1 \leq j \leq |Fr(B_i)|$. (Note that while generating the matrices, A's ID is skipped from $Fr(B_i)$.) Briefly,

B_i hashes each user in his/her friend list and sets the corresponding row entries to $Fr(B_i)[j]$ and 1 (indicating $Fr(B_i)[j]$ being B_i's friend) respectively. Then B_i encrypts his/her matrix M_i component-wise using the public key pk and sends it to A. Let the encrypted matrix be M_i'. Upon receiving the encrypted matrices, A performs homomorphic additions[†] on all the encrypted matrices, i.e., M_i''s component-wise and computes a new matrix Z, for $1 \leq j \leq s$:

$$\begin{aligned} Z[j][2] &\leftarrow \Pi_{i=1}^{m} M_i'[j][2] \\ Z[j][1] &\leftarrow Z[j][2]^{r_j} \cdot \Pi_{i=1}^{m} M_i'[j][1] \end{aligned}$$

where r_j is a random number chosen from $\mathbb{Z}_N^*$ and is used to randomize the encrypted user ID value. After that, A permutes the row vectors of Z by computing $\hat{Z} \leftarrow \pi(Z)$, where π is a random permutation function (known only to A) and sends it to T. Upon receiving $\hat{Z}$ from A, T decrypts the second component of each row of $\hat{Z}$ using his/her private key pr which gives the approximate frequency, denoted by $freq$, of corresponding hashed user(s). That is, T computes $freq_j = D_{pr}(\hat{Z}[j][2])$ and decrypts the first component of j^{th} row of $\hat{Z}$, for $1 \leq j \leq s$, as follows:

- If $freq_j \geq t$, decrypt the corresponding first component of $\hat{Z}$. That is, compute $D_{pr}(\hat{Z}[j][1])$ and set $\beta_j = \frac{D_{pr}(\hat{Z}[j][1])}{freq_j}$. Else, set β_j to 0.

- In case of no collision at location $\pi^{-1}(j)$ and $freq_j \geq t$, the observation is that $\beta_j = r_k + \mathrm{ID}_k$, where $k = \pi^{-1}(j)$ and ID_k is one of the recommended friend IDs to A. When $freq_j < t$, β_j is always 0.

- Sends β to A where $\beta = \langle \beta_1, \ldots, \beta_s \rangle$.

[†]Note that the multiplication and exponentiation operations on cipher-texts are followed by mod N^2 operation so that the resulting ciphertext is still in $\{0, 1, \ldots, N^2 - 1\}$. However, to make the presentation more clear, this work simply drops "mod N^2" from the computations in the rest of this section.

The decryption of the first component of each row of $\hat{Z}$ gives a random value. Finally, A applies the inverse permutation on vector β and removes the randomness for each non-zero entry. A selects the list of recommended friend IDs as shown below:

- Let vector F be $\pi^{-1}(\beta)$.
- $S = \{F_j - r_j \bmod N \mid F_j \in F \ \wedge \ F_j \neq 0, 1 \leq j \leq s\}$

Example 4. *Refer to Figure 5.1, let $t = 2$ and $s = 9$. From Figure 5.1, $Fr(Alex) = \langle Scott, Moore, Rice, Davis \rangle$. The goal is to recommend new friend(s) to Alex using the $PPFR_h$ protocol. Assume that the permutation function π and the hashed values of user IDs are as shown in Table 5.3. Let M_1, M_2, M_3 and M_4 denote the matrices generated by Scott, Moore, Rice and Davis respectively using the $PPFR_h$ protocol. The intermediate results during various steps involved in $PPFR_h$ are shown in Table 5.3. At the end of the $PPFR_h$ protocol, Baker and Martin are recommended as new friends to Alex.*

5.5.2. The PPFR$_{sp}$ Protocol. If s is large enough, the PPFR$_h$ protocol can produce very accurate results, but large s increases computation costs. To overcome this problem, this work presents an alternative PPFR protocol, termed as PPFR$_{sp}$, to accurately and efficiently recommend friends to A, at the expense of weaker security guarantee comparing to PPFR$_h$.

The overall steps involved in the PPFR$_{sp}$ protocol are demonstrated in Algorithm 7 and Algorithm 8. The main process of the protocol is presented in Algorithm 7. Assume that each user A in the network generates a public-private key pair (pk_A, pr_A) using the RSA public key system [94]. Also, each friend of an A's friend (say user C_{ij}) uses an AES encryption algorithm [58] to encrypt his/her data, and the secret key is divided into at most $|Fr(C)|$ different shares such that at least t shares are required to reconstruct the secret AES key [95].

Table 5.3: Intermediate results based on the PPFR$_h$ protocol

A	$Martin$	Bob	$Scott$	$Davis$	$Baker$	$Rice$	$Evans$	$Moore$	$Wilson$
$h_{a,b}(A)$	1	2	3	4	5	6	7	8	9
$\pi(h_{a,b}(A))$	5	8	1	6	9	4	3	7	2

$$M_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ Baker & 1 \\ 0 & 0 \\ 0 & 0 \\ Moore & 1 \\ 0 & 0 \end{pmatrix} ; \; M_2 = \begin{pmatrix} 0 & 0 \\ Bob & 1 \\ Scott & 1 \\ 0 & 0 \\ Baker & 1 \\ 0 & 0 \\ Evans & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} ; \; M_3 = \begin{pmatrix} Martin & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} ; \; M_4 = \begin{pmatrix} Martin & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ Wilson & 1 \end{pmatrix}$$

$$Z = \begin{pmatrix} E(2 \cdot (Martin + r_1)) & E(2) \\ E(Bob + r_2) & E(1) \\ E(Scott + r_3) & E(1) \\ E(0) & E(0) \\ E(2 \cdot (Baker + r_5)) & E(2) \\ E(0) & E(0) \\ E(Evans + r_7) & E(1) \\ E(Moore + r_8) & E(1) \\ E(Wilson + r_9) & E(1) \end{pmatrix} ; \quad \beta = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ Martin + r_1 \\ 0 \\ 0 \\ 0 \\ Baker + r_5 \end{pmatrix} ; \quad S = \{Baker, Martin\}$$

The main idea of the PPFR$_{sp}$ protocol is to let the candidates introduce themselves to user A in a privacy-preserving way. Here, the so called candidates are all the friends of A's friends. Suppose if B_i is one of A's friends and C_{ij} is B_i's friend, then C_{ij} acts as a candidate to be a new friend of A. The user A who wants some new friends just waits for the self introductions to come. First, B_i arranges a random path along which his/her friend (a candidate) will pass the self introduction to A. The random path can hide the identities of B_i's friends by preventing A from tracking back to them. The PPFR$_{sp}$ protocol is centered at user A to make friends of A's friends not aware of whom they are sending the self-introductions to. This preserves the privacy of both A and each user in $Fr(A)$. Also, since A cannot trace back to the candidate, the privacy of each candidate (i.e., friends of A's friends) is preserved.

Initially, A informs only his/her friends B_i that he/she wants new friends and wait for luck. Then, each B_i informs his/her friends the opportunity to make a new friend. To keep his/her friend list private from A, each B_i arranges the path of message passing for each user in $Fr(B_i)$ to hide the source [78], for $1 \leq i \leq m$. That is, B_i randomly picks two users X_{ij} and Y_{ij} for each of his/her friend C_{ij} and sends the path $M_{ij} = X_{ij}||Y_{ij}||E_{pu_{Y_{ij}}}(A)$ to C_{ij}. During the self-introduction part of the protocol, as shown in step 3 of Algorithm 7, C_{ij} receives the message and then appends $k_{C_{ij}}^{B_i}$ (the share of the key corresponding to B_i) and $\text{AES}_{k_{C_{ij}}}(C_{ij})$ (encrypted ID of C_{ij}) to M_{ij}, and sends only $Y_{ij}||E_{pu_{Y_{ij}}}(A)||k_{C_{ij}}^{B_i}||\text{AES}_{k_{C_{ij}}}(C_{ij})$ to X_{ij}. Then, X_{ij} forwards $E_{pu_{Y_{ij}}}(A)||k_{C_{ij}}^{B_i}||\text{AES}_{k_{C_{ij}}}(C_{ij})$ to Y_{ij} which decrypts the first part using his/her private key $pr_{Y_{ij}}$ to get A. After this, it forwards the remaining message $k_{C_{ij}}^{B_i}||\text{AES}_{k_{C_{ij}}}(C_{ij})$ to A. Finally, in the Collect_and_Solve part of the protocol, as shown in Algorithm 8, A waits and collects the returned messages from each Y_{ij} and groups them based on $\text{AES}_{k_{C_{ij}}}(C_{ij})$.

For each group G_{ij}, A proceeds as follows. If $|G_{ij}| \geq t$, A can generate the corresponding key ($k_{C_{ij}}$) from any t different partial keys in G_{ij} (using polynomial interpolation of t shares). Using $k_{C_{ij}}$, A can successfully decrypt the message $\text{AES}_{k_{C_{ij}}}(C_{ij})$ to get the user ID C_{ij} (newly recommended friend). Else, dump the group G_{ij} (since the number of different partial keys are less than t and A cannot generate the corresponding key). Note that the method in [95] guarantees that a group with size less than t will not provide enough information for A to generate the key, and consequently A cannot successfully decrypt $\text{AES}_{k_{C_{ij}}}(C_{ij})$. Observe that in the PPFR_{sp} protocol, since two random users are added to the path, all the messages received by A are sent to him/her by random users (namely Y_{ij}); therefore, A cannot trace back to the source C_{ij} that does the self-introduction. Note that two random users are necessary; otherwise, if only one random user were chosen to do the favor,

Algorithm 7 PPFR$_{sp}$

Require: $\forall$ user A, A has his/her RSA private key pr_A and shares his/her RSA public key pu_A to the public. A uses AES encryption algorithm to encrypt data and divides the secret AES key into at most $|Fr(A)|$ shares such that at least t shares are required to reproduce the secret AES key.

1: A:

(a). Send messages to each B_i that A wants some new friends where $B_i \in Fr(A)$

2: B_i $(1 \leq i \leq m)$:

(a). Receive message from A

(b). **for each** $C_{ij} \in Fr(B_i)$ **do**:

- Pick two random users X_{ij} and Y_{ij} from the social network
- $M_{ij} = X_{ij}||Y_{ij}||E_{pu_{Y_{ij}}}(A)$
- Send M_{ij} to C_{ij}

3: C_{ij} $(1 \leq i \leq m$ and $1 \leq j \leq |Fr(B_i)|)$:

(a). Receive M_{ij} from B_i

(b). **if** $|Fr(C_{ij})| \geq t$ **then**:

- Generate a share $k_{C_{ij}}^{B_i}$ of $k_{C_{ij}}$
- Send $Y_{ij}||E_{pu_{Y_{ij}}}(A)||k_{C_{ij}}^{B_i}||\text{AES}_{k_{C_{ij}}}(C_{ij})$ to X_{ij}

4: X_{ij} $(1 \leq i \leq m$ and $1 \leq j \leq |Fr(B_i)|$):

(a). Receive $Y_{ij}||E_{pu_{Y_{ij}}}(A)||k_{C_{ij}}^{B_i}||\text{AES}_{k_{C_{ij}}}(C_{ij})$ from C_{ij}

(b). Send $E_{pu_{Y_{ij}}}(A)||k_{C_{ij}}^{B_i}||\text{AES}_{k_{C_{ij}}}(C_{ij})$ to Y_{ij}

5: Y_{ij} $(1 \leq i \leq m$ and $1 \leq j \leq |Fr(B_i)|$):

(a). Receive $E_{pu_{Y_{ij}}}(A)||k_{C_{ij}}^{B_i}||\text{AES}_{k_{C_{ij}}}(C_{ij})$ from X_{ij}

(b). Decrypt $E_{pu_{Y_{ij}}}(A)$ to get A

(c). Send $k_{C_{ij}}^{B_i}||\text{AES}_{k_{C_{ij}}}(C_{ij})$ to A

6: A:

(a). Collect_and_Solve(t)

Algorithm 8 Collect_and_Solve(t)

1: $G = \{k_{C_{ij}}^{B_i} || \text{AES}_{k_{C_{ij}}}(C_{ij}), 1 \leq i \leq m \wedge 1 \leq j \leq |Fr(B_i)|\}$
2: Partition G into G_{ij}s, where
   $G_{ij} = \{k_{C_{ij}}^{B_{i_1}} || \text{AES}_{k_{C_{ij}}}(C_{ij}), \ldots, k_{C_{ij}}^{B_{i_z}} || \text{AES}_{k_{C_{ij}}}(C_{ij})\}$
3: $S = \emptyset$
4: **for** each G_{ij} **do**
5: **if** $|G_{ij}| \geq t$ **then**
6: Generate $k_{C_{ij}}$ via polynomial interpolation on
    $\{k_{C_{ij}}^{B_{i_1}}, \ldots, k_{C_{ij}}^{B_{i_z}}\}$
7: Using $k_{C_{ij}}$ to decrypt $\text{AES}_{k_{C_{ij}}}(C_{ij})$ to get C_{ij}
8: $S \leftarrow S \cup C_{ij}$
9: **end if**
10: **end for**
11: return S

he or she would know that C_{ij} and A share a friend. To increase security, one can choose more than two random users.

To protect the privacy of each $C_{ij} \in Fr(B_i)$, C_{ij} needs to encrypt his/her user ID (part of self-introduction), and attaches it with a partial key, which is done at step 3(b) of Algorithm 7. According to the theory proposed in [95], only the user that can collect at least a threshold (t) number of different partial keys can generate the full key to decrypt the user ID. Why is this necessary? Consider the goal of this protocol - only the users who share at least t friends with A should be recommended to A, A should not know anything else. Without the encryption of the self introduction, in addition to getting a few recommendations of new friends, A would know the user IDs of all his/her friends' friends which violates the privacy of B_i. Moreover, the AES secret key should be changed occasionally since it can be generated by A. Also note that only one partial key should correspond to one friend like B_i to ensure that A gets different partial keys from different friends shared by A and C_{ij}.

Example 5. *Consider the snapshot of the social network for Alex as shown in Figure 5.1, and let $t = 2$. Following from Figure 5.1, the set of potential candidates*

for Alex are $\langle Baker, Bob, Evans, Martin, Wilson\rangle$. *By applying the* $PPFR_{sp}$ *protocol, initially, Alex informs his friends* $\langle Scott, Moore, Rice, Davis\rangle$ *that he wants new friends. Then, each of Alex's friend will create an encrypted path for each of his/her friend. For example, Scott sends an encrypted path to his only friend Baker. Whereas, Moore creates three different encrypted paths and sends each one of them to her friends* $Baker, Bob$, *and Evans separately. Without loss of generality, let us consider the potential candidate Baker. Upon receiving the encrypted paths (different) from Scott and Moore, Baker appends his self introductions* $k_B^S||AES_{k_B}(Baker)$ *and* $k_B^M||AES_{k_B}(Baker)$ *to the respective random and encrypted paths. Where* k_B^S *and* k_B^M *are the corresponding key shares for Scott and Moore (only known to Baker). After this, Baker forwards the appended self-introduction messages along their respective random paths. Upon receiving the messages from random users, Alex groups the messages based on* $AES_{k_B}(Baker)$. *Since Alex has two different key shares i.e.,* k_B^S *and* k_B^M *corresponding to the group* $AES_{k_B}(Baker)$, *he generates the key* k_B *(as* $t = 2$*) and decrypts* $AES_{k_B}(Baker)$ *successfully and identifies Baker as the newly recommended friend. Similarly, Alex identifies Martin as a new friend.*

5.5.3. Complexity Analyses. Next, this work analyzes the computation and communication costs of the proposed protocols.

Computation Cost. Since the proposed protocols are asymmetric in nature, the computation costs vary for each party. For the PPFR_h protocol, the computation cost of each B_i is bounded by the number of encryption operations which depends on the hash domain size (s). Therefore, the computation complexity of B_i is bounded by $\text{O}(s)$ number of encryptions. (Note that each B_i performs the encryption operations independently using his/her friend list in parallel.) The computation complexity of A depends on the number of homomorphic addition and exponentiation (involved during randomization) operations. The number of homomorphic addition operations depends on s and $|Fr(A)|$. Whereas, the number of exponentiations depends on the

size of s. Therefore, the computation complexity of A is bounded by $O(s \cdot |Fr(A)|)$ homomorphic additions and $O(s)$ exponentiations. In addition, since T has to decrypt second component of each row of $\hat{Z}$, the computation complexity of T is bounded by $O(s)$ number of decryptions.

For the PPFR$_{sp}$ protocol, computation time mainly depends on the number of public key encryptions (which depends on the size of each B_i's friend list). Because the time taken to compute the AES private key, to generate the secret shares and to reconstruct the secret key is negligible comparing to the public key encryption costs. Therefore, the computation cost of the PPFR$_{sp}$ protocol is bounded by $O(|Fr(B)|)$ encryptions assuming that t and $|Fr(A)|$ are not exceedingly larger than $|Fr(B)|$.

Communication Cost. For the PPFR$_h$ protocol, the communication occurs between A and each of his/her friends, and also between A and T. Without loss of generality, let K denote the Paillier encryption key size (usually 1,024 bits long). The communication complexity between A and all of his/her friends is bounded by $O(K \cdot s \cdot |Fr(A)|)$ bits. In addition, A has to send the randomized encrypted matrix $\hat{Z}$ to T which is bounded by $O(K \cdot s)$ bits. Therefore, the overall communication complexity of PPFR$_h$ is bounded by $O(K \cdot s \cdot |Fr(A)|)$ bits.

In PPFR$_{sp}$, communication occurs among different users involved in each randomized path (i.e., $B_i \rightarrow C_{ij} \rightarrow X_{ij} \rightarrow Y_{ij} \rightarrow A$). Each B_i generates one path for each of his/her friend whose size is bounded by $O(K)$ bits. Then, the total communication between each B_i and his/her friends is bounded by $O(K \cdot |Fr(B_i)|)$ bits. Since there exist $\Sigma_{i=1}^{m}|Fr(B_i)|$ number of (B_i, C_{ij}) pairs, the total communication complexity of PPFR$_{sp}$ is bounded by $O(K \cdot l)$, where $l = \Sigma_{i=1}^{m}|Fr(B_i)|$. One disadvantage of the PPFR$_{sp}$ protocol is that it assumes that each X_{ij}, Y_{ij}, and C_{ij} are available and semi-honest, i.e., following the prescribed steps of the protocols.

5.5.4. Security Analysis. Next, a comprehensive analysis on the security of the proposed protocols is provided.

The Semi-Honest Security Model. There are two main reasons to adopt the semi-honest adversary model. First of all, a semi-honest model generally leads to more efficient secure protocols. Most existing practical secure protocols in secure multi-party computation (SMC) are under this model. By using standard zero-knowledge proofs [96] for threshold Paillier homomorphic encryption scheme [97, 98], the PPFR_h protocol can be easily converted into a secure protocol under the malicious model. However, its computation time is around 10 times more expensive than PPFR_h since zero-knowledge proofs are very costly.

More importantly, there is no need to design a secure protocol under the malicious model if the protocol is non-interactive. For example, the PPFR_h protocol only requires one round of communication between A and each user. There is no further interactions between A and each user. In the field of SMC, it is well-known open problem that input modification cannot be prevented [49]. That is, before executing a secure protocol, a participating party can provide any arbitrary input value to the protocol. There does not exist any mechanisms to determine if the input is legitimate. In addition, for a non-interactive protocol, any malicious behavior can be directly mapped to input modification. Since the proposed protocols are non-interactive, there is not need to enforce them to be secure under the malicious model.

Security Analysis under the Semi-Honest Model. To prove the security of the proposed protocols, this work adopts the standard proof methodology in the literature of SMC. According to Definition 1, one need to generate a simulated execution image of a protocol based on a participating party's private input and output. As long as the simulated execution image is computationally indistinguishable from the actual execution image, one can claim that the protocol is secure [49]. Since the proposed protocols are non-interactive, the security of participating parties is not symmetric. For instance, there is one way communication from B_i to A; therefore, A is considered as the adversary for each A's friend B_i. Similarly, T is considered as the adversary

for A. To prove PPFR$_h$ is secure, one need to show that the execution image of PPFR$_h$ from A's perspective (denoted by Π_A) does not leak any information regarding B_i's friend list. Also, it is required that the execution image of PPFR$_h$ from T's perspective (denoted by Π_T) does not leak any information regarding A's friend list.

Refer to the steps given in Algorithm 6, Π_A contains $\{h_{a,b}, pk, M'_1, \ldots, M'_m, S\}$. Then the simulated execution image of A (denoted by Π_A^S) can be generated as $\{h_{a,b}, pk, R_1, \ldots, R_m, S\}$, where $R_i[j][k]$ ($1 \leq j \leq s$ and $1 \leq k \leq 2$) is randomly chosen from $\mathbb{Z}_{N^2}$ and N is part of the public key pk. Since $M'_i[j][k]$ is an encrypted value generated from a semantically secure encryption scheme [55], $M'_i[j][k]$ is computationally indistinguishable from $R_i[j][k]$ [99]. As a result, Π_A is computationally indistinguishable from Π_A^S, and the PPFR$_h$ protocol is secure from each user B_i's perspective. That is, the friend list of B_i is not disclosed to A, except for what can be inferred from A's private input and output. Similarly, it is easy to prove that the PPFR$_h$ protocol is secure from each user A's perspective. That is, A's friend list is not disclosed to T. T only knows the frequency information of each permuted and randomized user IDs, and this is the only information leakage comparing to what can be achieved under the ideal trusted third party model. Also, since T does not know the hash function, T knows neither the friends of A nor any of the friends of A's friends. Therefore, the information leaked to T is negligible. In addition, nothing is revealed to A and B_i.

It is also possible to completely eliminate the information leakage to T by first adopting a fast secure binary conversion protocol over encrypted data [100] and comparing the result with the threshold t without ever disclosing the frequency information. In this way, the PPFR$_h$ offers the best security achievable in the semi-honest model. The price to pay here is the increased computation cost. According to the previous analysis, since the information leaked is extremely small, the security offered by the current implementation of PPFR$_h$ is likely sufficient.

In PPFR_{sp}, unlike PPFR_h where the scores are revealed to T, the common neighbors scores are directly revealed to A. However, A can only link the scores to the actual user IDs whose frequency is greater than or equal to t. If the common neighbors score is less than t, the corresponding real user ID is not revealed to A because the encrypted user ID cannot be decrypted by A according to the security guarantee of the Shamir's secret sharing scheme [95]. Furthermore, other than the intermediate encrypted results, no additional information is revealed to B_i. In general, PPFR_h is more secure than PPFR_{sp}, but it is less efficient.

Inference Problem and the Ideal Security Model. In the proposed protocols, there exists an inference problem, but it is worth pointing out that the inference problem exists even for the most secure common-neighbor based PPFR protocol. The reasoning is as follows.

In SMC, the security guarantee of a protocol is generally compared with the ideal security model: the trusted third party (TTP) model. Under the TTP model, there is a third party (P_0) who has the complete trust among all participating parties (e.g., A and $B_1, \ldots, B_m$ in the proposed protocols). Let PPFR_{P_0} denote a privacy-preserving friend recommendation protocol by utilizing a TTP. To implement PPFR_{P_0}, all participating parties send their private inputs to P_0, and P_0 will adopt the common neighbor method to identify potential new friends for A. At the end, P_0 will send the identified potential friends to A.

It is a well-known fact that a secure protocol implemented using a completely trusted party offers the maximum security guarantee [49]. However, when t is small, the inference problem, still exists even for the most secure protocol PPFR_{P_0}. E.g., when $t = m = 1$, A will learn the friend list of his or her only friend. The existence of the inference problem is not because PPFR_{P_0} is not secure, but it is mainly because the output of the common neighbor friend recommendation method reveals some additional information regarding the friend list of B_i. Therefore, under SMC, the

information that can be inferred from the output of a secure protocol is not considered as a security violation [49].

As illustrated in the ideal security model, to completely eliminate the inference problem is impossible. On the other hand, when the size of A's friend list m is two and $t = 1$, A will not be able to precisely know if the recommended friends are B_1 or B_2's friends since the recommended friends have equal probability for coming from either friend lists. For smaller values of t, the larger the m is, the more difficult for A to make correct inference. In general, the inference problem mainly depends on the size of m and t but not on the size of individual friend list. This is partly because A does not know the size of the friend lists of his or her friends and partly because the recommended friends are computed from an aggregated friend list whose contents are summed from B_1 to B_m's friend lists when $m \geq 2$.

To further mitigate the inference problem, the network service provider can fix t to a large number and makes it publicly available. Alternatively, if t is very small, each user B_i can refuse to participate to prevent the content of his or her friend list from being inferred.

5.6. EMPIRICAL ANALYSIS

This sub-section analyzes the effectiveness and computation cost of both protocols using a Facebook dataset. Since the PPFR_{sp} protocol always recommends friend(s) accurately, the effectiveness of PPFR_h is analyzed by using the PPFR_{sp} protocol as a baseline. In addition, the computation costs for both protocols are analyzed in detail.

5.6.1. Dataset and Platform Description. To conduct the experiments, friend lists of 11,500 Facebook users from Facebook.com are collected as follows. The crawler will first login using a Facebook account. It then starts to crawl the account's

friend list, the friend list of the friends in the list, and so on. To get a friend list of a Facebook account while logged in, the crawler will make use of the Facebook's AJAX API (a web service the Facebook web client side used to update its friend list display). The response will be a Javascript code file for the browser to update the web page, from where the friend list is extracted using a proper regular expression. Remember that Facebook lets users to set their privacy preferences. So one can only crawl those users who set their friend lists information as public.

Due to privacy settings, some of the extracted friend lists are empty (for 2,068 users). In addition, the combined friend lists of users' friends are empty for 112 users. Therefore, after removing the friend lists of these users, the experiments were conducted on the final set of friend lists for 9,330 users. Among the 9,330 users, the maximum and minimum friend list sizes are 447 and 1 respectively. In addition, the maximum number of unique friends of users' friends is 3,754, and each user has 24 friends on average. The proposed protocols were implemented in C, and experiments were performed on a Intel® Xeon® six-Core™ 3.07GHz PC running Ubuntu 10.04 with 12GB memory.

5.6.2. Effectiveness of PPFR$_h$. The effectiveness of PPFR$_h$ is analyzed by using PPFR$_{sp}$ as the baseline. Out of the 9,330 Facebook users, 1,000 users are randomly selected as target users and conducted various experiments based on different parameters. Let the target users (who wish to find new friends) be denoted by the set $\mathcal{D}$. Hereafter, the effectiveness results presented are average values over the 1,000 random users in $\mathcal{D}$. (The results are almost independent from the set and the size of the random users chosen when s is large enough. Also, different hash functions produce almost identical results.)

Let S_h and S_{sp} denote the sets of recommended friends resulting from PPFR$_h$ and PPFR$_{sp}$ respectively. Then, the accuracy of PPFR$_h$ (with respect to PPFR$_{sp}$) is

defined as the common ratio between the two sets as follows:

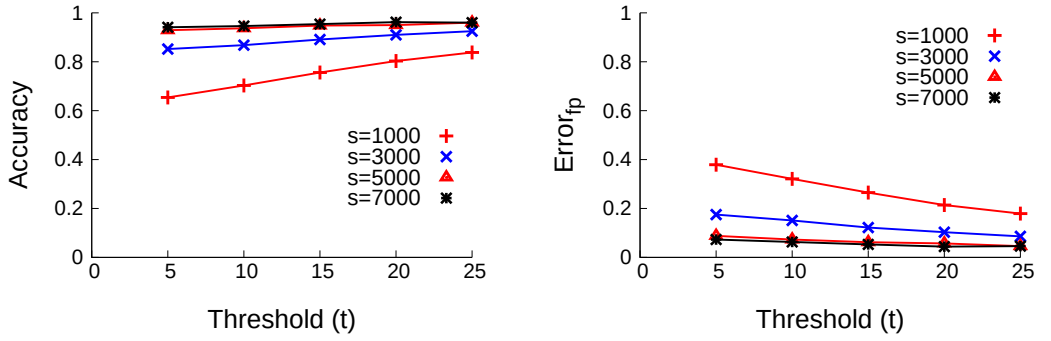
$$\text{Accuracy} = \frac{|S_h \cap S_{sp}|}{|S_{sp}|} \tag{5.4}$$

Note that S_{sp} always contains the accurate friend recommendations. The accuracy for each user in $\mathcal{D}$ is computed based on different hash/vector sizes (s), threshold values (t) and the results are as shown in Figure 5.2. Following from Figure 5.2(a), the accuracy of PPFR$_h$ is 65.4% when $s = 1,000$ and $t = 5$. Because the number of unique candidates for some of the users is greater than 1,000, this results in many collisions when $s = 1,000$. However, for $t = 5$, the accuracy increases from 65.4% to 94.1% as the value of s increases from 1,000 to 7,000. The reason behind that is as the value of s increases, the number of collisions are reduced; therefore, improving the accuracy. Also, as shown in Figure 5.2(a), for a fixed value of s, the accuracy improves with an increase in the value of t. For example, for $s = 5,000$, accuracy changes from 92.1% to 96% when t is varied from 5 to 25. A similar trend can be observed for other values of s and t. Collision rate of a universal hash function is generally low, and low collision rate introduces fewer false positives and false negatives when t is large.

Besides accuracy based on intersection size, the experiments also considered the false positive and false negative error rates for the PPFR$_h$ protocol which are defined below:

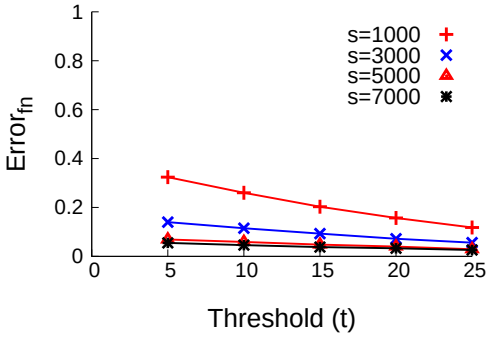
$$\text{Error}_{fp} = 1 - \frac{|S_{sp} \cap S_h|}{|S_h|}; \quad \text{Error}_{fn} = 1 - \frac{|S_{sp} \cap S_h|}{|S_{sp}|}$$

As shown in Figure 5.2(b), for $s = 1,000$ and $t = 5$, the false positive error is 37.9%. However, when t changes from 5 to 25, the false positive error drops to 17.9%. Whereas, for a fixed t, observe that the false positive error rate decreases when s increases (resulting in less hash collisions). For example, when $t = 25$, false

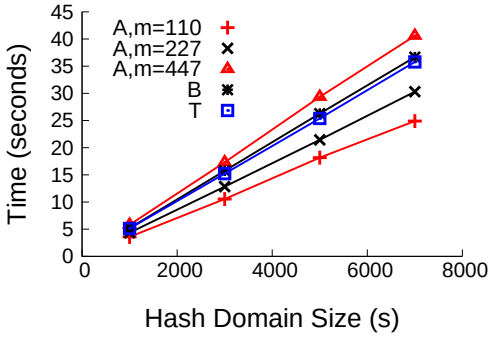


(a) Accuracy of PPFR_h

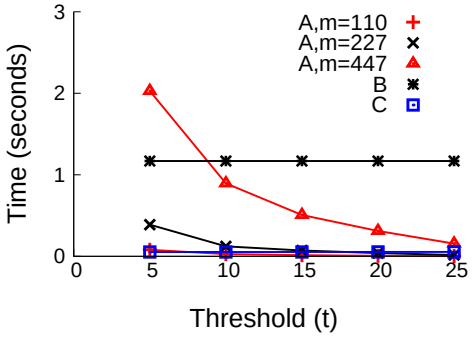
(b) False Positive Error



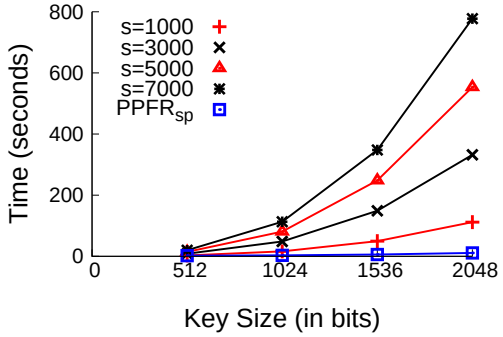
(c) False Negative Error



(d) Complexity: PPFR_h



(e) Complexity: PPFR_{sp}



(f) PPFR_h Vs. PPFR_{sp}

Figure 5.2: Performance evaluation of PPFR_h and PPFR_{sp}

positive error drops from 17.9% to 4.6% when s is increased from 1,000 to 7,000. A similar trend can be observed for false negative errors as shown in Figure 5.2(c).

Based on the above discussions, it is clear that PPFR_h gives good accuracy and low errors rates (both false positives and false negatives) when s ($\geq 3,000$) and

t (≥ 5). In particular, when $m = 7,000$ and $t = 25$, the PPFR$_h$ protocol recommends friends 96% accurately with a false positive error rate of 4.6% and a false negative error rate of 2.6%.

Next, the computation costs incurred on different parties in the PPFR$_h$ and PPFR$_{sp}$ protocols are analyzed separately. Also, the running time of both protocols for different values of s and key sizes are evaluated.

5.6.3. The Computation Cost of PPFR$_h$. For the PPFR$_h$ protocol, the computation costs are different for target user A, friend of A, and T. The computation cost of A depends on the size of his/her friend list ($m = |Fr(A)|$) and the hash domain or the vector size (s). Whereas, the computation cost of each friend of A (i.e., B) mainly depends on s (deciding factor for the number of encryptions to be performed by each B). For a friend B of A, observe that the cost involved in hashing and creating the matrix M is negligible comparing to the encryption cost involved in creating M'. Also, the computation cost of the party T is equivalent to the number of decryptions he/she needs to perform (which mainly depends on the values of s and t). In addition, for the PPFR$_h$ protocol, the computation costs of A and B are independent from t.

Based on the above discussions, the computation costs of each party in the PPFR$_h$ protocol are presented. Since the friend list sizes vary for different users, this work selected three different users with friend list sizes 110, 247, and 447 such that the size of one friend list is almost double the size of the other. This kind of selection is to purely present the variations in computation times in a more precise and concrete manner. However, similar results can be observed for other users. Note that in the above Facebook dataset, 447 is the maximum friend list size. For key size 1024, PPFR$_h$ is executed for each of these three users by varying the values of s.

As shown in Figure 5.2(d), for user A with $m = 110$, the computation time increases linearly with the value of s. When $m = 110$, the computation time of

A increases from 3.57 seconds to 10.54 seconds (increases by a factor of 3 due to expensive exponentiations) when s is changed from 1,000 to 3,000. A similar trend can be observed for the other values of s. On the other hand, when $m = 227$, the computation time of the user increases slightly (due to less expensive homomorphic additions) comparing to the user with $m = 110$ for a fixed value of s. In addition, as shown in Figure 5.2(d), the computation cost of B and T are the same under the assumption that T has to decrypt the whole encrypted matrix Z_1 (in the worst case).

Under the worst case scenario, the time taken for encrypting the matrix by B is almost the same as the time taken to decrypt the whole encrypted matrix by T. For instance, when $s = 5,000$, the computation time of B and T are 26.27 and 25.37 seconds (not depending on m), respectively. The above results indicate that the computation time incurred by the PPFR$_h$ protocol is more on B and T comparing to the time of A (except for the worst case where $m = 447$) irrespective of the hash domain size (s) because most operations performed at A are multiplications (much less costly than encryption or decryption operations).

5.6.4. The Computation Cost of PPFR$_{sp}$. The computation cost of PPFR$_{sp}$ depends on the running time of A, B, and C. On one hand, the computation cost of A depends on the number of introductions he/she receives and also on t. The computation cost of B depends on the number of randomized paths he/she creates (equivalent to the size of his/her friend list). On the other hand, the computation cost of C depends on the threshold t and $|Fr(C)|$ for generating shares of the AES secret key. In the experiments, Shamir's secret sharing scheme [95] is used to generate the secret partial keys or shares.

Consider the same three users as mentioned before. When A has 447 friends, he/she receives 13,444 self-introductions (through randomized paths). For $t = 5$, A consists of 461 groups (formed out of 13,444 self-introductions) and generates the AES secret keys for all groups in 2.02 seconds as shown in Figure 5.2(e). Similarly,

when $t = 25$, A has 37 groups and can generate the keys in 0.156 seconds. A similar trend can be observed for other two users. Note that since the values of t are small, the total time taken to generate all keys (following from the Lagrange Polynomial Interpolation) is very small. In the case of B, the computation cost only depends on his/her friend list and is independent of t. Therefore, as shown in Figure 5.2(e), it takes 1.169 seconds for B to generate all encrypted paths (assuming the worst case, where B can have 447 friends).

Since B performs public key encryptions, B takes more time comparing to A for t greater than or equal to 10 (resulting in fewer number of self-introductions). Additionally, the results are based on the worst case for computing C's time (i.e., assuming 447 friends). Therefore, C's computation time mainly depends on the time required to generate the secret shares based on t and its friend list size. Since the value of t is small, it takes 53 milliseconds for C to create all 447 secret shares and also to encrypt his/her ID using the AES private key encryption.

5.6.5. The Computation Cost of PPFR$_h$ vs. PPFR$_{sp}$. Finally, the total running times of both protocols for the worst case (i.e., A with 447 friends) with different key sizes and s values are as shown in Figure 5.2(f). It is clear that the computation time of PPFR$_h$ increases almost by a factor of 6 when the key size is doubled for any fixed value of s. For instance, if $s = 5,000$, the total run time of the PPFR$_h$ protocol increases from 80.853 seconds to 553.459 seconds when the key size is changed from 1,024 to 2,048 bits. The results show that PPFR$_{sp}$ is much faster than PPFR$_h$, when the key size is greater than or equal to 512 bits, by many orders of magnitude. However, PPFR$_h$ provides stronger security, as per the security definition of SMC [50, 77] comparing to the PPFR$_{sp}$ protocol.

A note on practicality: It is important to note that since friend recommendation needs not to be performed every minute or even every day, the proposed protocols are practical. The additional computation cost is a very small price to

pay comparing to the achieved privacy protection. More significantly, the proposed protocols make friend recommendation possible even when the friend lists are private.

5.7. IMPLEMENTATION DETAILS

This sub-section points out various implementation details involved for deploying the proposed protocols in real-world applications. In general, many online social networks (OSNs), such as Facebook, collect sensitive user profile data and store them on their server after proper encryption for security reasons. Therefore, users have no control over their stored data except trusting the OSN service and OSNs are free to share this stored data with third parties for business purposes. In order to give more control to users, this work considers the privacy-enhanced web pages where a user can encrypt his/her profile information before sending it to the OSN [57]. That is, the data stored on the OSN server is encrypted using user's secret key. When a user logins, his/her data stored on the OSN server is pushed back to the web page and decrypted. On the other hand, when the user tries to sign-out, his/her information is updated (in encrypted form) on the OSN server. Under this kind of architecture, can users still receive friend recommendations? Since T (i.e., the network administrator) has encrypted user's friend lists under different secret keys, it seems the process of friend recommendations by T alone is complex. Note that the friend list of each user is encrypted by using his/her secret key. In addition, the inherent information flow in OSNs makes the friend recommendation problem more challenging. However, by establishing a secure channel between the users, the proposed protocols make friend recommendation possible under the above mentioned architecture. In order to use the proposed protocols in a full fledge manner, one should first address an issue that arises due to the inherent information flow in OSNs which are discussed as below.

In the proposed protocols, users are assumed to exchange messages directly without revealing them to T (i.e., the network administrator). However, in many OSNs, messages are always passed through T because the user who is intendend to receive the message may not be online. Additionally, in order to minimize the information disclosure to T, one need to make sure that the messages exchanged between users are in encrypted form and only the intended receiver should be able to decrypt it. The above issue can be solved by creating a secure session between the users. Briefly, assume that each user holds a public/private key pair, where the public keys are treated as global information. If user B wants to send some message to A, then B generates an AES encryption key (i.e., the session key which should be different from the secret key used to encrypt and store his/her data on the OSN server), encrypts it using the public key of A, and forwards it to A through T. Note that here T merely acts as a network router that simply forwards the message to A. Upon receiving the encrypted message, A decrypts it to get the AES session key which is used for further secure communication with B.

Once this kind of secure session is established between users, the proposed protocols can be directly applied where two users A and B can communicate securely by simply encrypting their messages using the AES session key and forwarding them to one another through T. Note that the AES session key should be changed occasionally for security reasons. The proposed protocols, after taking the above implementation details into account, protect the friend list of a user from other users and T.

5.8. CONCLUSION

The emerging growth of social networks has resulted in vast amount of social data which need to be mined for various kinds of recommendations such as friend

recommendation. However, since the social data contain personal and sensitive information about individual users, when friend lists remain private, existing friend recommendation techniques do not work. As a result, this section proposed two privacy preserving friend recommendation algorithms under the assumption that users' friend lists are private. Both of the proposed protocols recommend new friends using the common neighbors proximity measure in a privacy-preserving manner.

The first protocol PPFR_h is based on an additive homomorphic encryption scheme, and its accuracy is essentially based on the parameters of universal hash function $h_{a,b}$. Whereas the second protocol PPFR_{sp} utilizes the concept of protecting the source privacy through randomizing the message passing path and also recommends friends accurately. The observation is that the PPFR_{sp} protocol is more efficient than the PPFR_h protocol. The proposed PPFR protocols act as a trade-off among security, efficiency and accuracy.

For the PPFR_{sp} protocol, if both A and his/her friends have long friend lists, then the protocol will cause a large number of communication between the users. In general, only a small number of users are recommended to A; thus, most of the messages passing back to A will end up useless. One possible solution is to develop a pruning mechanism so that some of the candidates could be pruned during the process. Another issue is that both of the proposed protocols are designed only to compute the recommendation in a nearest neighbor way. On the other hand, a new friend can also be recommended in other ways using various information, like the similarities of sharing contents, user profiles or activities in the network. As a future work, it will be interesting to explore alternative ways for developing hybrid privacy-preserving friend recommendation methods by combining different scoring functions.

6. PRIVACY-AWARE FRIEND RECOMMENDATIONS IN OUTSOURCED SOCIAL NETWORKS

As an emerging computing paradigm, cloud computing attracts many organizations to consider utilizing the benefits of a cloud in terms of cost-efficiency, flexibility, and offload of administrative overhead. With cloud computing, users now have the opportunity to outsource their data as well as the data management services to the cloud [101, 102]. On one hand, by outsourcing, the organization gets the benefit of reducing the data management costs and improves the quality of service. On the other hand, hosting and query processing of data out of the organization's control raises security challenges such as preserving data confidentiality. Therefore, due to the rise of various privacy issues, sensitive data (e.g., user's profile information) need to be encrypted before outsourcing to the cloud. In addition, all computations should be handled by the cloud; otherwise, there would be no point to outsource the data at the first place.

One straightforward way to protect the confidentiality of the outsourced data from the cloud as well as from the unauthorized users is to encrypt data by the data owner before outsourcing [103]. By this way, the data owner can protect the privacy of his/her own data. This work assumes that users profile data were encrypted and then outsourced to the cloud. In general, performing computations over encrypted data without the cloud ever decrypting the data is a very challenging task. This work focuses on solving the friend recommendation problem over encrypted users' profiles outsourced to a cloud using k-nearest neighbor (kNN) method. That is, given the encrypted profile of a target user (denoted by input query Q), the goal is for the cloud to securely identify the k-nearest user profiles to Q. The question here is how can the cloud execute a set of operations over encrypted data while the data stored at the cloud are encrypted at all times? In the literature, various techniques related

to query processing over encrypted data have been proposed, including range queries [104, 105, 106] and other aggregate queries [107, 108]. However, these techniques are either not applicable or inefficient to solve advanced queries such as the k-nearest neighbor (kNN) query.

From the above discussion, it is clear that friend recommendations under outsourced social networking environment is closely related to the problem of secure k-nearest neighbor (denoted by SkNN) query. Briefly, given the query Q of a target user, the objective of the SkNN problem is to securely identify the k-nearest profiles to Q using the database of encrypted profiles (say T) in the cloud, without allowing the cloud to learn anything regarding the actual contents of the database T and the query Q. More specifically, assuming encrypted data are outsourced to a cloud, an effective SkNN protocol needs to satisfy the following properties:

- Preserve the confidentiality of T and Q at all times
- Hiding data access patterns from the cloud
- Accurately compute the k-nearest neighbors of query Q
- Incur low computation overhead on the end-users

In the past few years, researchers have proposed various methods [101, 109, 110, 111] to address the SkNN problem. However, the existing SkNN methods violate at least one of the above mentioned desirable properties of a SkNN protocol. On one hand, the methods in [101, 109] are insecure because they are vulnerable to chosen and known plaintext attacks. On the other hand, recent method in [111] returns non-accurate kNN result to the end-user. More precisely, in [111], the cloud retrieves the relevant encrypted partition instead of finding the encrypted exact k-nearest neighbors. Furthermore, in [101, 110, 111], the end-user involves in heavy computations during the query processing step. By doing so, these methods utilize cloud as just a storage medium, i.e., no significant work is done on the cloud side.

Additionally, the existing SkNN methods do not protect data access patterns from the cloud. More details regarding the existing SkNN methods are provided in Section 6.3.

This work first presents a basic scheme to solve SkNN and demonstrates that such a naive solution is not secure. To provide better security, a novel secure kNN protocol is also proposed that protects both confidentiality of the data and access patterns to the data. Also, this work empirically analyzes the efficiency of the proposed protocols through various experiments. The results indicate that the secure protocol is very efficient on the user end, and this lightweight scheme allows a user to use any mobile device to request friend recommendations. The protocols developed in this paper are secure under the semi-honest model [50]. However, they can be easily extended to secure protocols under other adversary models, such as malicious and covert, using threshold based cryptosystem and zero-knowledge proofs.

6.1. PROBLEM DEFINITION

In the outsourced environment, assume the existence of two non-colluding semi-honest cloud service providers, denoted by C_1 and C_2, which together form a federated cloud. Such an assumption is not new and has been commonly used in the related problem domains [112, 113]. The intuition behind such an assumption is as follows. Most of the cloud service providers in the market are well-established IT companies, such as Amazon and Google. Therefore, a collusion between them is highly unlikely as it will damage their reputation which in turn effects their revenues. Also, assume that C_2 generates a pair of public-secret key pair (pk, sk) based on an additive homomorphic public-key cryptosystem that is semantically secure (e.g., Paillier cryptosystem [55]).

Let us assume that a user profile can be represented as a vector based on a pre-determined set of attributes. Here the attributes can be location, education, hobbies, friend list and so on. In the outsourced social networking environment, this work assumes that each user encrypts his/her profile vector attribute-wise using the public key pk of C_2 and sends the encrypted vector along with his/her user ID to C_1. Now consider the problem of recommending friends to a target user Bob by C_1. For this purpose, C_1 randomly picks a subset of n users whose profile vectors are denoted by $T = \{t_1, \ldots, t_n\}$. Note that C_1 has encrypted version of T, say $E_{pk}(T)$, where $E_{pk}(.)$ denotes the encryption function of an additively homomorphic public-key cryptosystem that is semantically secure. Let $Q \notin T$ be the profile vector of Bob such that $Q = \langle q_1, \ldots, q_m \rangle$, where m denotes the number of attributes. Then, the goal is for C_1 and C_2 to jointly compute the k-nearest neighbors of Q who are recommended as potential friends to Bob. During this process, Bob's query Q and contents of database T should not be revealed to C_1 and C_2. In addition, the access patterns to data (e.g., the user IDs corresponding to top k profiles for any given Q) should be protected from the clouds. Such a process is referred to as Secure kNN (SkNN) query over encrypted data in the cloud. Without loss of generality, let $\langle t'_1, \ldots, t'_k \rangle$ denote the k-nearest profiles to Q. Then, the SkNN protocol can be formally defined as follows:

$$\text{S}k\text{NN}(E_{pk}(T), E_{pk}(Q)) \rightarrow \langle \text{ID}(t'_1), \ldots, \text{ID}(t'_k) \rangle$$

where $\text{ID}(t'_i)$ denotes the user ID of profile t'_i. Note that, at the end of the SkNN protocol, the output $\langle \text{ID}(t'_1), \ldots, \text{ID}(t'_k) \rangle$ should be revealed only to Bob.

6.2. MAIN CONTRIBUTIONS

This work proposes a novel SkNN protocol to facilitate friend recommendations in outsourced social networks based on the k-nearest neighbor search over encrypted data in the cloud that protects user's privacy and hides access patterns. In the proposed protocol, once the encrypted data are outsourced to the cloud, users do not participate in any computations. In particular, the proposed protocol meets the following requirements:

- **Data confidentiality -** Contents of T or any intermediate results should not be revealed to the clouds.
- **Query privacy -** Bob's input query Q should not be revealed to the clouds.
- **Correctness -** The output $\langle \mathrm{ID}(t'_1), \ldots, \mathrm{ID}(t'_k) \rangle$ should be computed accurately and revealed only to Bob. Additionally, no information other than $\mathrm{ID}(t'_1), \ldots, \mathrm{ID}(t'_k)$ should be revealed to Bob.
- **Low computation overhead on Bob -** The proposed protocols incur low computation overhead on Bob compared with the existing works [101, 109, 110, 111].
- **Hidden data access patterns -** Access patterns to the data, such as the profile IDs corresponding to the k-nearest neighbors of Q, should not be revealed to the clouds (to prevent any inference attacks).

It is worth pointing out that the intermediate results seen by the clouds in the proposed protocol are either newly generated randomized encryptions or random numbers. Thus, which data profiles correspond to the k-nearest neighbors of Q are not known to the clouds. Also, Bob does not involve in any computations. Hence, data access patterns are further protected from Bob.

6.3. RELATED WORK AND BACKGROUND

This sub-section presents an overview of the existing secure k-nearest neighbor techniques. Then, it discusses the security definition adopted in this paper along with the homomorphic properties of the Paillier cryptosystem as a background.

6.3.1. Existing SkNN Techniques. Retrieving the k-nearest neighbors to a given query Q is one of the most fundamental problem in many application domains such as similarity search, pattern recognition, and data mining. In the literature, many techniques have been proposed to address the SkNN problem, which can be classified into two categories based on whether the data are encrypted or not: *centralized* and *distributed*.

Centralized Methods. In the centralized methods, the data owner is assumed to outsource his/her database and DBMS functionalities (e.g., kNN query) to an untrusted external service provider which manages the data on behalf of the data owner where only trusted users are allowed to query the hosted data. By outsourcing data to an untrusted server, many security issues arise, such as data privacy (protecting the confidentiality of the data from the server and query issuer). To achieve data privacy, data owner is required to use data anonymization models (e.g., k-anonymity) or cryptographic (e.g., encryption and data perturbation) techniques over his/her data before outsourcing them to the server.

Encryption is a traditional technique used to protect the confidentiality of sensitive data such as medical records. Due to data encryption, the process of query evaluation over encrypted data becomes challenging. Along this direction, various techniques have been proposed for processing range [104, 105, 106] and aggregation queries [107, 108] over encrypted data. However, this work restricts the discussion to secure evaluation of kNN query.

In the past few years, researchers have proposed different methods [101, 109, 110, 111] to address the SkNN problem. Wong et al. [109] proposed a new encryption scheme called asymmetric scalar-product-preserving encryption (ASPE) that preserves scalar product between the query vector Q and any tuple vector t_i from database T for distance comparison which is sufficient to find kNN. In [109], data and query are encrypted using slightly different encryption schemes before outsourcing to the server and all the query users know the decryption key. As an improvement, Zhu et al. [110] proposed a novel SkNN method in which the key of the data owner is not disclosed to the user. However, their architecture requires the participation of data owner during query encryption. As an alternative, Hu et al. [101] proposed a method based on provably secure privacy homomorphism encryption scheme from [114] that supports modular addition, subtraction and multiplication over encrypted data. They addressed the SkNN problem under the following setting: the client has the ciphertexts of all data points in database T and the encryption function of T whereas the server has the decryption function of T and some auxiliary information regarding each data point. However, both methods in [101, 109] are not secure because they are vulnerable to chosen-plaintext attacks. Also, all the above methods leak data access patterns to the server.

Recently, Yao et al. [111] proposed a new SkNN method based on partition-based secure Voronoi diagram (SVD). Instead of asking the cloud to retrieve the exact kNN, they required, from the cloud, to retrieve a relevant encrypted partition $E_{pk}(G)$ for $E_{pk}(T)$ such that G is guaranteed to contain the k-nearest neighbors of Q. However, this work solves the SkNN problem accurately by letting the cloud to retrieve the exact k-nearest neighbors of Q (in encrypted form). In addition, most of the computations during the query processing step in [101, 110, 111] are performed locally by the end-user which conflicts the very purpose of outsourcing the DBMS

functionalities to the cloud. Furthermore, the protocol in [111] leaks data access patterns, such as the partition ID corresponding to a user query, to the cloud.

Data Distribution Methods. In the data distributed methods, data are assumed to be partitioned either vertically or horizontally and distributed among a set of independent, non-colluding parties. In the literature, the data distributed methods rely on secure multiparty computation (SMC) techniques that enable multiple parties to securely evaluate a function using their respective private inputs without disclosing the input of one party to the others. Many efforts have been made to address the problem of kNN query in a distributed environment. Shaneck et al. [115] proposed privacy-preserving algorithm to perform k-nearest neighbor search. The protocol in [115] is based on secure multiparty computation for privately computing kNN points in a horizontally partitioned dataset. Qi et al. [116] proposed a single-step kNN search protocol that is provably secure with linear computation and communication complexities. Vaidya et al. [117] studied privacy-preserving top-k queries in which the data are vertically partitioned. Ghinita et al. [118] proposed a private information retrieval (PIR) based framework for answering kNN queries in location-based services. In [118], the data residing at the server are in plaintext format. However, if the data are encrypted to ensure data confidentiality, it is not clear how a user can obliviously retrieve the output records because he/she does not know the indexes that match his/her input query. Nevertheless, even if a user can retrieve the records using PIR, the user still needs to perform local computations to identify the k-nearest neighbors. However, in the framework proposed in this section, the users computation is completely outsourced to a cloud.

In summary, the above data distribution methods are not applicable to perform kNN queries over encrypted data for two reasons: (1). This work deals with encrypted form of database and query which is not the case in the above methods (2). The database in this work is assumed to be encrypted and stored on the cloud

whereas in the above methods it is partitioned (in plaintext format) among different parties. Some common notations that are used extensively in this paper are shown in Table 6.1.

6.3.2. Security Definition. In this paper, privacy/security is closely related to the amount of information disclosed during the execution of a protocol. There are many ways to define information disclosure. To maximize privacy or minimize information disclosure, this work adopts the security definitions in the literature of secure multiparty computation (SMC) first introduced by Yao's Millionaires' problem for which a provably secure solution was developed [38, 39]. In this work, the participating parties are assumed to be semi-honest; that is, a semi-honest party follows the rules of the protocol using its correct input, but is free to later use what it sees during execution of the protocol to compromise security. Specific details regarding the semi-honest security definition adopted in this work is given in Section 2. We also emphasize that more details on standard security definitions and models can be found in [50].

6.3.3. Paillier Cryptosystem. The Paillier cryptosystem is an additive homomorphic and probabilistic asymmetric encryption scheme [55]. Let E_{pk} be the encryption function with public key pk given by (N, g), where N is a product of two large primes and g is in $\mathbb{Z}^*_{N^2}$. Also, let D_{sk} be the decryption function with secret key sk. For any given plaintexts $a, b \in \mathbb{Z}_N$, the Paillier encryption scheme exhibits the following properties:

a. **Homomorphic Addition -** $E_{pk}(a + b) \leftarrow E_{pk}(a) * E_{pk}(b) \bmod N^2$;

b. **Homomorphic Multiplication -** $E_{pk}(a * b) \leftarrow E_{pk}(a)^b \bmod N^2$;

c. **Semantic Security -** The encryption scheme is semantically secure [56], i.e., given a set of ciphertexts, an adversary cannot deduce any information about the plaintext.

Table 6.1: Common notations used in the SkNN protocols

T	A subset of user profiles
$E_{pk}(T)$	Attribute-wise encryption of T
n	Number of user profiles in T
m	Number of attributes in the profile vectors
t_i	i^{th} profile vector in T
Q	Bob's profile vector
t'_i	i^{th} nearest profile vector to Q based on T
ID(t'_i)	User ID corresponding to t'_i
l	Domain size (in bits) of the squared Euclidean distance between profile vectors
$\langle z_1, z_l \rangle$	The most and least significant bits of integer z
$[z]$	Vector of encryptions of the individual bits of z

This work assumes that a data owner or a social network user encrypted his or her data using Paillier cryptosystem before outsourcing them to a cloud.

6.4. BASIC SECURITY PRIMITIVES

This sub-section presents a set of generic protocols that will be used as sub-routines while constructing the proposed SkNN protocol in Section 6.6. All of the below protocols are considered under two-party semi-honest setting. In particular, assume the existence of two semi-honest parties P_1 and P_2 such that the Paillier's secret key sk is known only to P_2 whereas pk is treated as public.

- Secure Multiplication (SM) Protocol:

  This protocol considers P_1 with input $(E_{pk}(a), E_{pk}(b))$ and outputs $E_{pk}(a * b)$ to P_1, where a and b are not known to P_1 and P_2. During this process, no information regarding a and b is revealed to P_1 and P_2. The output $E_{pk}(a * b)$ is known only to P_1.

- Secure Squared Euclidean Distance (SSED) Protocol:

  P_1 with input $(E_{pk}(X), E_{pk}(Y))$ and P_2 securely compute the encryption of squared Euclidean distance between vectors X and Y. Here X and Y are two m dimensional vectors given by $E_{pk}(X) = \langle E_{pk}(x_1), \ldots, E_{pk}(x_m)\rangle$ and $E_{pk}(Y) = \langle E_{pk}(y_1), \ldots, E_{pk}(y_m)\rangle$. At the end of SM, the output $E_{pk}(|X - Y|^2)$ is known only to P_1.

- Secure Bit-Decomposition (SBD) Protocol:

  P_1 with input $E_{pk}(z)$ and P_2 securely compute the encryptions of the individual bits of z, where $0 \leq z < 2^l$. The output $[z] = \langle E_{pk}(z_1), \ldots, E_{pk}(z_l)\rangle$ is known only to P_1. Here z_1 and z_l denote the most and least significant bits of integer z respectively.

- Secure Minimum (SMIN) Protocol:

  P_1 with input $([u], [v])$ and P_2 with sk securely compute the encryptions of the individual bits of minimum number between u and v. That is, the output is $[\min(u, v)]$ which will be known only to P_1. During this protocol, no information regarding u and v is revealed to P_1 and P_2.

- Secure Minimum out of n Numbers (SMIN_n) Protocol:

  In this protocol, P_1 has n encrypted vectors $([d_1], \ldots, [d_n])$ and P_2 has sk. Here $[d_i] = \langle E_{pk}(d_{i,1}), \ldots, E_{pk}(d_{i,l})\rangle$ such that $d_{i,1}$ and $d_{i,l}$ are the most and least significant bits of integer d_i respectively, for $1 \leq i \leq n$. P_1 and P_2 jointly compute the output $[\min(d_1, \ldots, d_n)]$. At the end, $[\min(d_1, \ldots, d_n)]$ is known only to P_1. During SMIN_n, no information about d_i's is revealed to P_1 and P_2.

- Secure Bit-OR (SBOR) Protocol:

  P_1 with input $(E_{pk}(o_1), E_{pk}(o_2))$ and P_2 securely compute $E_{pk}(o_1 \vee o_2)$, where o_1 and o_2 are two bits. The output $E_{pk}(o_1 \vee o_2)$ is known only to P_1.

Next, each of these protocols are discussed in detail. Also, this work either proposes a new solution or refers to the most efficient known implementation to each one of them.

6.4.1. Secure Multiplication (SM). Consider that P_1 holds private input $(E_{pk}(a), E_{pk}(b))$ and P_2 holds the secret key sk. The goal of the secure multiplication (SM) protocol is to return the encryption of $a * b$, i.e., $E_{pk}(a * b)$ as output to P_1. During this protocol, no information regarding a and b is revealed to P_1 and P_2. The basic idea of SM is based on the following property which holds for any given $a, b \in \mathbb{Z}_N$:

$$a * b = (a + r_a) * (b + r_b) - a * r_b - b * r_a - r_a * r_b \tag{6.1}$$

where all the arithmetic operations are performed under $\mathbb{Z}_N$. The overall steps in SM are shown in Algorithm 9. Briefly, P_1 initially randomizes a and b by computing $a' = E_{pk}(a) * E_{pk}(r_a)$ and $b' = E_{pk}(b) * E_{pk}(r_b)$, and sends them to P_2. Here r_a and r_b are random numbers in $\mathbb{Z}_N$ known only to P_1. Upon receiving, P_2 decrypts and multiplies them to get $h = (a + r_a) * (b + r_b) \bmod N$. Then, P_2 encrypts h and sends it to P_1. After this, P_1 removes extra random factors from $h' = E_{pk}((a+r_a)*(b+r_b))$ based on Equation 6.1 to get $E_{pk}(a * b)$. Note that, for any given $x \in \mathbb{Z}_{\mathbb{N}}$, "$N - x$" is equivalent to "$-x$" under $\mathbb{Z}_N$. Hereafter, the notation $r \in_R \mathbb{Z}_N$ is used to denote r as a random number in $\mathbb{Z}_N$.

Example 6. *Suppose* $a = 59$ *and* $b = 58$. *For simplicity, let* $r_a = 1$ *and* $r_b = 3$. *Initially,* P_1 *computes* $a' = E_{pk}(60) = E_{pk}(a) * E_{pk}(r_a)$, $b' = E_{pk}(61) = E_{pk}(b) * E_{pk}(r_b)$ *and sends them to* P_2. *Then,* P_2 *decrypts and multiplies them to get* $h = 3660$. *After this,* P_2 *encrypts* h *to get* $h' = E_{pk}(3660)$ *and sends it to* P_1. *Upon receiving* h', P_1 *computes* $s = E_{pk}(3483) = E_{pk}(3660 - a * r_b)$, *and* $s' = E_{pk}(3425) = E_{pk}(3483 - b * r_a)$. *Finally,* P_1 *computes* $E_{pk}(a * b) = E_{pk}(3422) = E_{pk}(3425 - r_a * r_b)$. □

Algorithm 9 SM$(E_{pk}(a), E_{pk}(b)) \rightarrow E_{pk}(a * b)$

Require: P_1 has $E_{pk}(a)$ and $E_{pk}(b)$; P_2 has sk

1: P_1:

(a). Pick two random numbers $r_a, r_b \in \mathbb{Z}_N$

(b). $a' \leftarrow E_{pk}(a) * E_{pk}(r_a)$

(c). $b' \leftarrow E_{pk}(b) * E_{pk}(r_b)$; send a', b' to P_2

2: P_2:

(a). Receive a' and b' from P_1

(b). $h_a \leftarrow D_{sk}(a')$ and $h_b \leftarrow D_{sk}(b')$

(c). $h \leftarrow h_a * h_b \bmod N$

(d). $h' \leftarrow E_{pk}(h)$

(e). Send h' to P_1

3: P_1:

(a). Receive h' from P_2

(b). $s \leftarrow h' * E_{pk}(a)^{N-r_b}$

(c). $s' \leftarrow s * E_{pk}(b)^{N-r_a}$

(d). $E_{pk}(a * b) \leftarrow s' * E_{pk}(r_a * r_b)^{N-1}$

6.4.2. Secure Squared Euclidean Distance (SSED). Suppose P_1 holds two encrypted vectors $(E_{pk}(X), E_{pk}(Y))$ and P_2 holds sk. Here X and Y are two m-dimensional vectors such that $E_{pk}(X) = \langle E_{pk}(x_1), \ldots, E_{pk}(x_m)\rangle$ and $E_{pk}(Y) = \langle E_{pk}(y_1), \ldots, E_{pk}(y_m)\rangle$. The goal of SSED is to securely compute $E_{pk}(|X-Y|^2)$, where $|X - Y|$ denotes the Euclidean distance between X and Y. During this protocol, no information regarding X and Y is revealed to P_1 and P_2. The basic idea of the SSED protocol follows from the following equation:

$$|X - Y|^2 = \sum_{i=1}^{m} (x_i - y_i)^2 \tag{6.2}$$

Algorithm 10 SSED$(E_{pk}(X), E_{pk}(Y)) \rightarrow E_{pk}(|X-Y|^2)$

Require: P_1 has $E_{pk}(X)$ and $E_{pk}(Y)$; P_2 has sk

1: P_1, **for** $1 \leq i \leq m$ **do**:

(a). $E_{pk}(x_i - y_i) \leftarrow E_{pk}(x_i) * E_{pk}(y_i)^{N-1}$

2: P_1 and P_2, **for** $1 \leq i \leq m$ **do**:

(a). Compute $E_{pk}((x_i - y_i)^2)$ using the SM protocol

3: P_1 computes $E_{pk}(|X-Y|^2) \leftarrow \prod_{i=1}^{m} E_{pk}((x_i - y_i)^2)$

The main steps involved in SSED are shown in Algorithm 10. Briefly, for $1 \leq i \leq m$, P_1 initially computes $E_{pk}(x_i - y_i)$ by using the homomorphic properties. Then P_1 and P_2 jointly compute $E_{pk}((x_i - y_i)^2)$ using the SM protocol, for $1 \leq i \leq m$. Note that the outputs of SM are known only to P_1. By applying homomorphic properties on $E_{pk}((x_i - y_i)^2)$, P_1 computes $E_{pk}(|X-Y|^2)$ locally based on Equation 6.2.

Example 7. *Suppose P_1 holds two encrypted data records $E_{pk}(X) = \langle E_{pk}(63), E_{pk}(1), E_{pk}(1), E_{pk}(145), E_{pk}(233), E_{pk}(1), E_{pk}(3), E_{pk}(0), E_{pk}(6), E_{pk}(0)\rangle$ whereas $E_{pk}(Y) = \langle E_{pk}(56), E_{pk}(1), E_{pk}(3), E_{pk}(130), E_{pk}(256), E_{pk}(1), E_{pk}(2), E_{pk}(1), E_{pk}(6), E_{pk}(2)\rangle$. During the SSED protocol, P_1 initially computes $E_{pk}(x_1 - y_1) = E_{pk}(7), \ldots, E_{pk}(x_{10} - y_{10}) = E_{pk}(-2)$. Then, P_1 and P_2 jointly compute $E_{pk}((x_1 - y_1)^2) = E_{pk}(49) = SM(E_{pk}(7), E_{pk}(7)), \ldots, E_{pk}((x_{10} - y_{10})^2) = SM(E_{pk}(-2), E_{pk}(-2)) = E_{pk}(4)$. P_1 locally computes $E_{pk}(|X-Y|^2) = E_{pk}(\sum_{i=1}^{10}(x_i - y_i)^2) = E_{pk}(813)$.* □

6.4.3. Secure Bit-Decomposition (SBD). Assume that P_1 has $E_{pk}(z)$ and P_2 has sk, where z is not known to both parties and $0 \leq z < 2^l$. The goal of the secure bit-decomposition (SBD) protocol is to compute the encryptions of the individual bits of binary representation of z [100]. That is, the output is $[z] = \langle E_{pk}(z_1), \ldots, E_{pk}(z_l)\rangle$, where z_1 and z_l denote the most and least significant bits of z respectively. At the end, the output $[z]$ is known only to P_1. Since the goal of this

work is not to investigate the existing SBD protocols, the most efficient SBD scheme that was recently proposed in [100] is used in the proposed protocols.

Example 8. *Let us suppose that* $z = 55$ *and* $l = 6$. *Then the SBD protocol with private input* $E_{pk}(55)$ *gives* $[55] = \langle E_{pk}(1), E_{pk}(1), E_{pk}(0),\ E_{pk}(1), E_{pk}(1), E_{pk}(1)\rangle$ *as the output to* P_1. □

6.4.4. Secure Minimum (SMIN). In this protocol, P_1 with input $([u], [v])$ and P_2 with secret key sk securely compute the encryptions of the individual bits of $\min(u, v)$, i.e., the output is $[\min(u, v)]$. $[u] = \langle E_{pk}(u_1), \ldots, E_{pk}(u_l)\rangle$ and $[v] = \langle E_{pk}(v_1), \ldots, E_{pk}(v_l)\rangle$, where u_1 (resp., v_1) and u_l (resp., v_l) are the most and least significant bits of u (resp., v). At the end of SMIN, the output $[\min(u, v)]$ is known only to P_1.

By assuming that $0 \leq u, v < 2^l$, this work proposes a novel SMIN protocol. The basic idea of the proposed SMIN protocol is for P_1 to randomly choose the functionality F (by flipping a coin), where F is either $u > v$ or $v > u$, and to obliviously execute F with P_2. Since F is randomly chosen and known only to P_1, the output of the functionality F is oblivious to P_2. Based on the output and chosen F, P_1 computes $[\min(u, v)]$ locally using homomorphic properties.

The overall steps involved in the SMIN protocol are shown in Algorithm 11. To start with, P_1 initially chooses the functionality F as either $u > v$ or $v > u$ randomly. Then, using SM, P_1 computes $E_{pk}(u_i * v_i)$ with the help of P_2. Now, depending on F, P_1 proceeds as follows, for $1 \leq i \leq l$:

- If $F : u > v$, compute

$$W_i = E_{pk}(u_i) * E_{pk}(u_i * v_i)^{N-1} = E_{pk}(u_i * (1 - v_i))$$
$$\Gamma_i = E_{pk}(v_i - u_i) * E_{pk}(\hat{r}_i) = E_{pk}(v_i - u_i + \hat{r}_i)$$

Algorithm 11 SMIN$([u],[v]) \rightarrow [\min(u,v)]$

Require: P_1 has $[u]$ and $[v]$, where $0 \leq u, v < 2^l$; P_2 has sk

1: P_1:

(a). Randomly choose the functionality F

(b). **for** $i = 1$ to l **do**:

- $E_{pk}(u_i * v_i) \leftarrow \text{SM}(E_{pk}(u_i), E_{pk}(v_i))$
- **if** $F : u > v$ **then**:
  - $W_i \leftarrow E_{pk}(u_i) * E_{pk}(u_i * v_i)^{N-1}$
  - $\Gamma_i \leftarrow E_{pk}(v_i - u_i) * E_{pk}(\hat{r}_i)$; $\hat{r}_i \in_R \mathbb{Z}_N$

  else
  - $W_i \leftarrow E_{pk}(v_i) * E_{pk}(u_i * v_i)^{N-1}$
  - $\Gamma_i \leftarrow E_{pk}(u_i - v_i) * E_{pk}(\hat{r}_i)$; $\hat{r}_i \in_R \mathbb{Z}_N$
- $G_i \leftarrow E_{pk}(u_i \oplus v_i)$
- $H_i \leftarrow H_{i-1}^{r_i} * G_i$; $r_i \in_R \mathbb{Z}_N$ and $H_0 = E_{pk}(0)$
- $\Phi_i \leftarrow E_{pk}(-1) * H_i$
- $L_i \leftarrow W_i * \Phi_i^{r'_i}$; $r'_i \in_R \mathbb{Z}_N$

(c). $\Gamma' \leftarrow \pi_1(\Gamma)$ and $L' \leftarrow \pi_2(L)$; send Γ' and L' to C

2: P_2:

(a). Receive Γ' and L' from P_1

(b). $M_i \leftarrow D_{sk}(L'_i)$, for $1 \leq i \leq l$

(c). **if** $\exists\ j$ such that $M_j = 1$ **then** $\alpha \leftarrow 1$
else $\alpha \leftarrow 0$

(d). $M'_i \leftarrow \Gamma'^{\alpha}_i$, for $1 \leq i \leq l$; send M' and $E_{pk}(\alpha)$ to P_1

3: P_1:

(a). Receive M' and $E_{pk}(\alpha)$ from P_2; compute $\widetilde{M} \leftarrow \pi_1^{-1}(M')$

(b). **for** $i = 1$ to l **do**:

- $\lambda_i \leftarrow \widetilde{M}_i * E_{pk}(\alpha)^{N-\hat{r}_i}$
- **if** $F : u > v$ **then** $E_{pk}(\min(u,v)_i) \leftarrow E_{pk}(u_i) * \lambda_i$
  else $E_{pk}(\min(u,v)_i) \leftarrow E_{pk}(v_i) * \lambda_i$

- If $F : v > u$, compute:

$$W_i = E_{pk}(v_i) * E_{pk}(u_i * v_i)^{N-1} = E_{pk}(v_i * (1 - u_i))$$
$$\Gamma_i = E_{pk}(u_i - v_i) * E_{pk}(\hat{r}_i) = E_{pk}(u_i - v_i + \hat{r}_i)$$

  where $\hat{r}_i$ is a random number in $\mathbb{Z}_N$ known only to P_1.

- Observe that if $F : u > v$, then $W_i = E_{pk}(1)$ only if $u_i > v_i$, and $W_i = E_{pk}(0)$ otherwise. Similarly, when $F : v > u$, $W_i = E_{pk}(1)$ only if $v_i > u_i$, and $W_i = E_{pk}(0)$ otherwise. Also, depending of F, Γ_i stores the encryption of randomized difference between u_i and v_i which will be used in later computations.

- Compute the encrypted bit-wise XOR between the bits u_i and v_i as $G_i = E_{pk}(u_i \oplus v_i)$ using the formulation $G_i = E_{pk}(u_i) * E_{pk}(v_i) * E_{pk}(u_i * v_i)^{N-2}$. For any two bits o_1 and o_2, the property $o_1 \oplus o_2 = o_1 + o_2 - 2(o_1 * o_2)$ always holds.

- Compute an encrypted vector H by preserving the first occurrence of $E_{pk}(1)$ (if there exists one) in G by initializing $H_0 = E_{pk}(0)$. The rest of the entries of H are computed as $H_i = H_{i-1}^{r_i} * G_i$. Note that at most one of the entry in H is $E_{pk}(1)$ and the remaining entries are encryptions of either 0 or a random number. Also, if there exists an index j such that $H_j = E_{pk}(1)$, then index j is the first position (from the most significant bit) at which the corresponding bits of u and v differ.

- Then, P_1 computes $\Phi_i = E_{pk}(-1) * H_i$. Note that "$-1$" is equivalent to "$N-1$" under $\mathbb{Z}_N$. From the above discussions, it is clear that $\Phi_i = E_{pk}(0)$ at most once since H_i is equal to $E_{pk}(1)$ at most once. Also, if $\Phi_j = E_{pk}(0)$, then index j is the position at which the bits of u and v differ first.

- Compute an encrypted vector L by combining W and Φ. Note that W_i stores the result of $u_i > v_i$ or $v_i > u_i$ which depends on F known only to P_1. Precisely, P_1 computes $L_i = W_i * \Phi_i^{r_i'}$, where r_i' is a random number in $\mathbb{Z}_N$. The observation here is if $\exists$ an index j such that $\Phi_j = E_{pk}(0)$, denoting the first flip in the bits of u and v, then W_j stores the corresponding desired information, i.e., whether $u_j > v_j$ or $v_j > u_j$ in encrypted form.

After this, P_1 permutes the encrypted vectors Γ and L using two random permutation functions π_1 and π_2. Specifically, P_1 computes $\Gamma' = \pi_1(\Gamma)$ and $L' = \pi_2(L)$, and sends them to P_2. Upon receiving, P_2 decrypts L' component-wise to get $M_i = D_{sk}(L_i')$, for $1 \leq i \leq l$, and checks for index j (decide the output of F). That is, if $M_j = 1$, then the output of F is 1, and 0 otherwise. Let the output be α. Note that since F is not known to P_2, the output α is oblivious to P_2. In addition, P_2 computes a new encrypted vector M' where $M_i' = \Gamma_i'^{\alpha}$, for $1 \leq i \leq l$, sends M' and $E_{pk}(\alpha)$ to P_1. After receiving M' and $E_{pk}(\alpha)$, P_1 computes the inverse permutation of M' as $\widetilde{M} = \pi_1^{-1}(M')$. Then, P_1 performs the following homomorphic operations to compute the encryption of i^{th} bit of $\min(u, v)$, i.e., $E_{pk}(\min(u, v)_i)$, for $1 \leq i \leq l$:

- Remove the randomness from $\widetilde{M_i}$ by computing

$$\lambda_i = \widetilde{M_i} * E_{pk}(\alpha)^{N - \hat{r}_i}$$

- If $F : u > v$, compute the i^{th} encrypted bit of $\min(u, v)$ as $E_{pk}(\min(u, v)_i) = E_{pk}(u_i) * \lambda_i = E_{pk}(u_i + \alpha * (v_i - u_i))$. Otherwise, compute $E_{pk}(\min(u, v)_i) = E_{pk}(v_i) * \lambda_i = E_{pk}(v_i + \alpha * (u_i - v_i))$.

In the SMIN protocol, one main observation (upon which the correctness of the final output can also be justified) is that if $F : u > v$, then $\min(u, v)_i = (1 - \alpha) * u_i + \alpha * v_i$ always holds, for $1 \leq i \leq l$. Similarly, if $F : v > u$, then $\min(u, v)_i = \alpha * u_i + (1 - \alpha) * v_i$ always holds.

Example 9. *Consider that* $u = 55$, $v = 58$, *and* $l = 6$. *In addition, assume that* P_1*'s random permutation functions are given as below. Suppose that* P_1 *holds* $[u] = [55] =$

i	$=$	1	2	3	4	5	6
		$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
$\pi_1(i)$	$=$	6	5	4	3	2	1
$\pi_2(i)$	$=$	2	1	5	6	3	4

$\langle E_{pk}(1), E_{pk}(1), E_{pk}(0), E_{pk}(1), E_{pk}(1), E_{pk}(1)\rangle$ *and* $[v] = [58] = \langle E_{pk}(1), E_{pk}(1), E_{pk}(1), E_{pk}(0), E_{pk}(1), E_{pk}(0)\rangle$. *Without loss of generality, let us assume that* P_1 *chooses the functionality* $F : v > u$. *Then, various intermediate results based on the SMIN protocol are as shown in Table 6.2. Following from Table 6.2, observe that:*

- *At most one of the entry in* H *is* $E_{pk}(1)$ *(*$= H_3$*) and the remaining entries are encryptions of either 0 or a random number in* $\mathbb{Z}_N$. *Index* $j = 3$ *is the first position at which the corresponding bits of* u *and* v *differ.*

- $\Phi_3 = E_{pk}(0)$ *since* H_3 *is equal to* $E_{pk}(1)$. *Also, since* $M_5 = 1$, P_2 *sets* α *to 1.*

At the end, only P_1 *knows* $[\min(u, v)] = [u] = [55]$. □

6.4.5. Secure Minimum out of n Numbers (SMIN$_n$). Consider P_1 with input $([d_1], \ldots, [d_n])$ and P_2 with sk, where $[d_i] = \langle E_{pk}(d_{i,1}), \ldots, E_{pk}(d_{i,l})\rangle$ and $0 \leq d_i < 2^l$, for $1 \leq i \leq n$. The goal of the SMIN$_n$ protocol is to compute $[\min(d_1, \ldots, d_n)] = [d_{\min}]$ without revealing any information about d_i's to P_1 and P_2. This work constructs a new SMIN$_n$ protocol by utilizing SMIN as the building block. The proposed SMIN$_n$ protocol is an iterative approach and it computes the desired output in an hierarchical fashion. In each iteration, minimum between a pair of values is computed and are feeded as input to the next iteration. Therefore, generating a binary execution tree in a bottom-up fashion. At the end, only P_1 knows the final result $[d_{\min}]$.

The overall steps involved in the proposed SMIN$_n$ protocol are highlighted in Algorithm 12. Initially, P_1 assigns $[d_i]$ to a temporary vector $[d'_i]$, for $1 \leq i \leq n$. Also,

Table 6.2: Example of SMIN where $F : v > u$, $u = 55$ and $v = 58$

$[u]$	$[v]$	W_i	Γ_i	G_i	H_i	Φ_i	L_i	Γ'_i	L'_i	M_i	λ_i	$\min_i$
1	1	0	r	0	0	-1	r	$1+r$	r	r	0	1
1	1	0	r	0	0	-1	r	r	r	r	0	1
0	1	1	$-1+r$	1	1	0	1	$1+r$	r	r	-1	0
1	0	0	$1+r$	1	r	r	r	$-1+r$	r	r	1	1
1	1	0	r	0	r	r	r	r	1	1	0	1
1	0	0	$1+r$	1	r	r	r	r	r	r	1	1

*All column values are in encrypted form ($E_{pk}(.)$) except M_i column. Also, r is a random in $\mathbb{Z}_N$ which is different for each row and column.

he/she creates a global variable *num* and initialize it to n, where *num* represents the number of (non-zero) vectors involved in each iteration. Since the SMIN$_n$ protocol executes in a binary tree hierarchy (bottom-up fashion), it has $\lceil \log_2 n \rceil$ iterations, and in each iteration, the number of vectors involved varies. In the first iteration (i.e., $i = 1$), P_1 with private input $([d'_{2j-1}], [d'_{2j}])$ and P_2 with sk involve in the SMIN protocol, for $1 \leq j \leq \lfloor \frac{num}{2} \rfloor$. At the end of the first iteration, only P_1 knows $[\min(d'_{2j-1}, d'_{2j})]$ and nothing is revealed to P_2, for $1 \leq j \leq \lfloor \frac{num}{2} \rfloor$. Also, P_1 stores the result $[\min(d'_{2j-1}, d'_{2j})]$ in $[d'_{2j-1}]$, updates $[d'_{2j}]$ to zero and *num* to $\lceil \frac{num}{2} \rceil$.

During the i^{th} iteration, only the non-zero vectors are involved, for $2 \leq i \leq \lceil \log_2 n \rceil$. For example, during second iteration (i.e., $i = 2$), only $[d'_1], [d'_3]$, and so on are involved. Note that in each iteration, the output is revealed only to P_1 and *num* is updated to $\lceil \frac{num}{2} \rceil$. At the end of SMIN$_n$, P_1 assigns the final encrypted binary vector of global minimum value, i.e., $[\min(d_1, \ldots, d_n)]$ which is stored in $[d'_1]$ to $[d_{\min}]$.

Example 10. *For example, assume that P_1 holds $\langle [d_1], \ldots, [d_6] \rangle$ (i.e., $n = 6$). Then, based on the SMIN$_n$ protocol, the binary execution tree (in a bottom-up fashion) to*

Algorithm 12 $\text{SMIN}_n([d_1], \ldots, [d_n]) \rightarrow [d_{\min}]$

Require: P_1 has $([d_1], \ldots, [d_n])$; P_2 has sk

1: P_1:

(a). $[d'_i] \leftarrow [d_i]$, for $1 \leq i \leq n$

(b). $num \leftarrow n$

2: P_1 and P_2, **for** $i = 1$ to $\lceil \log_2 n \rceil$ **do**:

(a). **for** $1 \leq j \leq \left\lfloor \frac{num}{2} \right\rfloor$ **do**:

- **if** $i = 1$ **then**:
  - $[d'_{2j-1}] \leftarrow \text{SMIN}([d'_{2j-1}], [d'_{2j}])$
  - $[d'_{2j}] \leftarrow 0$

  else
  - $[d'_{2i(j-1)+1}] \leftarrow \text{SMIN}([d'_{2i(j-1)+1}], [d'_{2ij-1}])$
  - $[d'_{2ij-1}] \leftarrow 0$

(b). $num \leftarrow \left\lceil \frac{num}{2} \right\rceil$

3: P_1 sets $[d_{\min}]$ to $[d'_1]$

compute $[\min(d_1, \ldots, d_6)]$ *is as shown in Figure 6.1. Note that,* $[d'_i]$ *is initially set to* $[d_i]$*, for* $1 \leq i \leq 6$. □

6.4.6. Secure Bit-OR (SBOR). P_1 holds $(E_{pk}(o_1), E_{pk}(o_2))$ and P_2 holds sk, where o_1 and o_2 are two bits not known to both parties. The goal of the SBOR protocol is to securely compute $E_{pk}(o_1 \vee o_2)$. At the end of this protocol, only P_1 knows $E_{pk}(o_1 \vee o_2)$. During this process, no information related to o_1 and o_2 is revealed to P_1 and P_2. Using SM, P_1 and P_2 compute $E_{pk}(o_1 \vee o_2)$ as follows:

- P_1 with input $(E_{pk}(o_1), E_{pk}(o_2))$ and P_2 with sk involve in the SM protocol. At the end of this step, the output $E_{pk}(o_1 * o_2)$ is known only to P_1. Note that, since o_1 and o_2 are bits, $E_{pk}(o_1 * o_2) = E_{pk}(o_1 \wedge o_2)$.

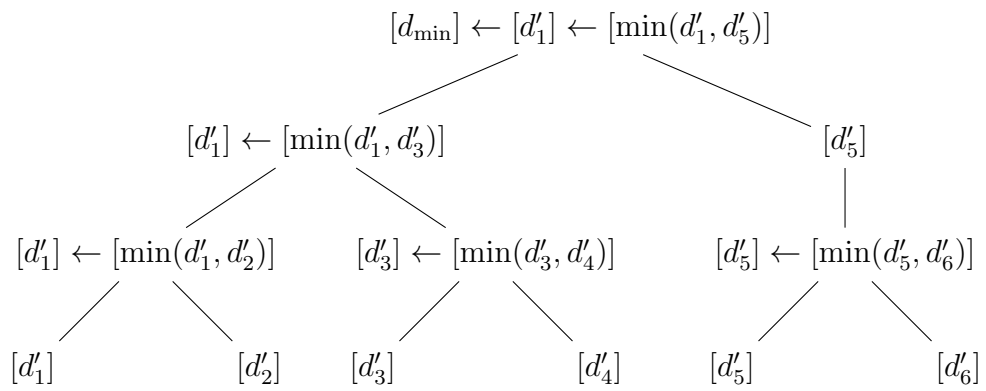


Figure 6.1: Binary execution tree for $n = 6$ based on the SMIN_n protocol

- $E_{pk}(o_1 \vee o_2) = E_{pk}(o_1 + o_2) * E_{pk}(o_1 \wedge o_2)^{N-1}$.

In general, for any given two bits o_1 and o_2, the property $o_1 \vee o_2 = o_1 + o_2 - o_1 \wedge o_2$ always holds.

6.5. SECURITY ANALYSIS OF BASIC PRIMITIVES UNDER THE SEMI-HONEST MODEL

First of all, it is worth pointing out that the outputs in the above mentioned protocols are always in encrypted format, and are known only to P_1. In addition, all the intermediate results revealed to P_2 are either random or pseudo-random. Note that, the SBD protocol proposed in [100] is secure under the semi-honest model.

Since the proposed SMIN protocol (which is used as a sub-routine in SMIN_n) is more complex than other protocols mentioned above, this work provides its security proof rather than providing proofs for each protocol. Therefore, only a formal security proof for the SMIN protocol is included in this work based on the standard simulation argument [50]. Nevertheless, similar proof strategies can be used to show that other protocols are secure under the semi-honest model. Informally speaking, this work claims that all the intermediate results seen by P_1 and P_2 in the mentioned protocols are either random or pseudo-random.

Proof of Security for SMIN: As mentioned in Section 2, to formally prove that SMIN is secure [50] under the semi-honest model, one need to show that the simulated execution image of SMIN is computationally indistinguishable from the actual execution image of SMIN. An execution image generally includes the messages exchanged and the information computed from these messages. Therefore, according to Algorithm 11, let the execution image of P_2 be denoted by $\Pi_{P_2}(\text{SMIN})$, where

$$\Pi_{P_2}(\text{SMIN}) = \{\langle \Gamma'_i, \mu_i + \hat{r}_i \bmod N\rangle, \langle L'_i, \alpha\rangle \mid \text{for } 1 \leq i \leq l\}$$

Observe that $\mu_i + \hat{r}_i \bmod N$ is derived upon decrypting Γ'_i, where the modulo operator is implicit in the decryption function. Also, P_2 receives L' from P_1 and let α denote the (oblivious) comparison result computed from L'. Without loss of generality, suppose the simulated image of P_2 be $\Pi^S_{P_2}(\text{SMIN})$, where

$$\Pi^S_{P_2}(\text{SMIN}) = \{\langle s'_{1,i}, s'_{2,i}\rangle, \langle s'_{3,i}, \alpha'\rangle \mid \text{for } 1 \leq i \leq l\}$$

Here $s'_{1,i}$ and $s'_{3,i}$ are randomly generated from $\mathbb{Z}_{N^2}$, and $s'_{2,i}$ is randomly generated from $\mathbb{Z}_N$. In addition, α' is a random bit. Since E_{pk} is a semantically secure encryption scheme with resulting ciphertext size less than N^2, Γ'_i and L'_i are computationally indistinguishable from $s'_{1,i}$ and $s'_{3,i}$, respectively. Also, as $\hat{r}_i$ is randomly generated, $\mu_i + \hat{r}_i \mod N$ is computationally indistinguishable from $s'_{2,i}$. Furthermore, because the functionality is randomly chosen by P_1 (at step 1(a) of Algorithm 11), α is either 0 or 1 with equal probability. Thus, α is computationally indistinguishable from α'. Combining all these results together, this work concludes that $\Pi_{P_2}(\text{SMIN})$ is computationally indistinguishable from $\Pi^S_{P_2}(\text{SMIN})$. This implies that during the execution of SMIN, P_2 does not learn any information regarding u, v and the actual comparison result. Intuitively speaking, the information P_2 has during an execution of SMIN is either random or pseudo-random, so this information does not disclose

anything regarding u and v. Additionally, as F is known only to P_1, the actual comparison result is oblivious to P_2.

On the other hand, the execution image of P_1, denoted by $\Pi_{P_1}(\text{SMIN})$, can be given by

$$\Pi_{P_1}(\text{SMIN}) = \{M'_i, E_{pk}(\alpha) \mid \text{for } 1 \leq i \leq l\}$$

Here M'_i is an encrypted value, which is random in $\mathbb{Z}_{N^2}$, received from P_2 (at step 3(a) of Algorithm 11). Let the simulated image of P_1 be $\Pi^S_{P_1}(\text{SMIN})$, where

$$\Pi^S_{P_1}(\text{SMIN}) = \{s'_{4,i}, b \mid \text{for } 1 \leq i \leq l\}$$

Both $s'_{4,i}$ and b are randomly generated from $\mathbb{Z}_{N^2}$. Since E_{pk} is a semantically secure encryption scheme with resulting ciphertext size less than N^2, M'_i and $E_{pk}(\alpha)$ are computationally indistinguishable from $s_{4,i}$ and b. Therefore, $\Pi_{P_1}(\text{SMIN})$ is computationally indistinguishable from $\Pi^S_{P_1}(\text{SMIN})$. This implies that P_1 cannot learn any information regarding u, v and the comparison result during the execution of SMIN.

Based on the above analysis, it is clear that the proposed SMIN protocol is secure under the semi-honest model. In a similar way, one can formally prove that all the protocols given in the previous section are secure under the semi-honest model. Hence, in the rest of this section, assume that the basic primitives presented in Section 6.4 are secure under the semi-honest model.

6.6. THE PROPOSED SkNN PROTOCOLS

This sub-section first presents a basic SkNN protocol and demonstrates why such a simple solution is not secure. Then, it discusses the second approach, a fully secure kNN protocol. Both protocols are constructed using the security primitives discussed in the previous sub-section as building blocks.

As mentioned earlier, this work assumes the existence of two non-colluding semi-honest cloud service providers C_1 and C_2 which together form a federated cloud. Also, assume that user profile vectors were encrypted by individual users (using the public key of C_2) and sent to C_1. Now the goal of SkNN is to recommend k potential friends to the target user Bob using k-nearest neighbors technique. For this purpose, assume C_1 selects a random set of n user profiles denoted by $T = \{t_1, \ldots, t_n\}$. Let the encrypted user profiles of T be denoted by $E_{pk}(T)$. Also, assume that all attribute values and their Euclidean distances lie in $[0, 2^l)$. Note that the secret key sk is known only to C_2.

The proposed SkNN protocols retrieve the top k user profiles that are closest to the Bob's query Q in an efficient and secure manner. At a high level, C_1 and C_2 involve in a set of sub-protocols to securely retrieve (in encrypted form) the set of k profiles corresponding to the k-nearest neighbors of the input profile Q. At the end of the proposed protocols, only Bob will receive the user IDs of k-nearest neighbors to Q as the output.

6.6.1. The Basic Protocol. In the basic secure k-nearest neighbor query protocol, denoted by SkNN$_\mathrm{b}$, the desirable properties are relaxed to produce an efficient protocol (more details are given in the later part of this section).

The main steps involved in the SkNN$_\mathrm{b}$ protocol are given in Algorithm 13. Initially, C_1 with private input $(E_{pk}(Q), E_{pk}(t_i))$ and C_2 with the secret key sk jointly involve in the SSED protocol, where $E_{pk}(t_i) = \langle E_{pk}(t_{i,1}), \ldots, E_{pk}(t_{i,m})\rangle$, for $1 \leq i \leq n$. The output of this step, denoted by $E_{pk}(d_i)$, is the encryption of squared Euclidean distance between Q and t_i, i.e., $d_i = |Q - t_i|^2$. As mentioned earlier, $E_{pk}(d_i)$ is known only to C_1, for $1 \leq i \leq n$. Note that computation of exact Euclidean distance between encrypted vectors is hard to achieve as it involves square root. However, in the k-nearest neighbor problem, it is sufficient to compare the squared Euclidean distances as it preserves relative ordering. After this, C_1 sends $\{\langle 1, E_{pk}(d_1)\rangle, \ldots, \langle n, E_{pk}(d_n)\rangle\}$

Algorithm 13 $\mathrm{S}k\mathrm{NN}_\mathrm{b}(E_{pk}(T), E_{pk}(Q)) \rightarrow \langle \mathrm{ID}(t'_1), \ldots, \mathrm{ID}(t'_k)\rangle$

Require: C_1 has $E_{pk}(T)$ and $E_{pk}(Q)$; C_2 has sk

1: C_1 and C_2:

(a). **for** $i = 1$ to n **do**:

- $E_{pk}(d_i) \leftarrow \mathrm{SSED}(E_{pk}(Q), E_{pk}(t_i))$

(b). C_1 sends $\{\langle 1, E_{pk}(d_1)\rangle, \ldots, \langle n, E_{pk}(d_n)\rangle\}$ to C_2

2: C_2:

(a). Receive $\{\langle 1, E_{pk}(d_1)\rangle, \ldots, \langle n, E_{pk}(d_n)\rangle\}$ from C_1

(b). $d_i \leftarrow D_{sk}(E_{pk}(d_i))$, for $1 \leq i \leq n$

(c). Generate $\delta \leftarrow \langle i_1, \ldots, i_k\rangle$, such that $\langle d_{i_1}, \ldots, d_{i_k}\rangle$ are the top k smallest distances among $\langle d_1, \ldots, d_n\rangle$

(d). Send δ to C_1

3: C_1:

(a). Receive δ from C_2

(b). $\mathrm{ID}(t'_j) \leftarrow \mathrm{ID}(t_{i_j})$, for $1 \leq j \leq k$

(c). Send $\langle \mathrm{ID}(t'_1), \ldots, \mathrm{ID}(t'_k)\rangle$ to Bob

to C_2, where entry $\langle i, E_{pk}(d_i)\rangle$ correspond to data record t_i, for $1 \leq i \leq n$. Upon receiving $\langle 1, E_{pk}(d_1)\rangle, \ldots, \langle n, E_{pk}(d_n)\rangle$, C_2 decrypts the encrypted distance in each entry to get $d_i = D_{sk}(E_{pk}(d_i))$. Then, C_2 generates an index list $\delta = \langle i_1, \ldots, i_k\rangle$ such that $\langle d_{i_1}, \ldots d_{i_k}\rangle$ are the top k smallest distances among $\langle d_1, \ldots, d_n\rangle$. After this, C_2 sends δ to C_1. Upon receiving δ, C_1 simply sets $\mathrm{ID}(t'_j)$ to $\mathrm{ID}(t_{i_j})$, for $1 \leq j \leq k$, and recommends $\langle \mathrm{ID}(t'_1), \ldots, \mathrm{ID}(t'_1)\rangle$ as top k potential friends to Bob.

6.6.2. Fully Secure kNN Protocol. The above-mentioned $\mathrm{S}k\mathrm{NN}_\mathrm{b}$ protocol reveals the data access patterns to C_1 and C_2. That is, for any given Q, C_1 and C_2 know which data records correspond to the k-nearest neighbors of Q. Also, it reveals d_i values to C_2 and top k profile IDs to C_1. However, leakage of such information may not be acceptable. Along this direction, a fully secure protocol, denoted

by SkNN$_{\rm m}$ (where m stands for maximally secure), is proposed here to retrieve the k-nearest neighbors of Q. The proposed SkNN$_{\rm m}$ protocol preserves all the desirable properties of a secure kNN protocol as mentioned in the Introduction.

The main steps involved in the proposed SkNN$_{\rm m}$ protocol are as shown in Algorithm 14. Initially, C_1 with private input $(E_{pk}(Q), E_{pk}(t_i))$ and C_2 with the secret key sk jointly involve in the SSED protocol. The output of this step is $E_{pk}(d_i) = E_{pk}(|Q - t_i|^2)$ which will be known only to C_1, for $1 \leq i \leq n$. Then, C_1 with input $E_{pk}(d_i)$ and C_2 with sk securely compute the encryptions of the individual bits of d_i using the SBD protocol. Note that the output of this step $[d_i] = \langle E_{pk}(d_{i,1}), \ldots, E_{pk}(d_{i,l})\rangle$ is known only to C_1, where $d_{i,1}$ and $d_{i,l}$ are the most and least significant bits of d_i respectively. Note that $0 \leq d_i < 2^l$, for $1 \leq i \leq n$.

After this, C_1 and C_2 compute the top k (in encrypted form) closest profiles that are closest to Q in an iterative manner. More specifically, they compute $E_{pk}(\mathrm{ID}(t'_1))$ in the first iteration, $E_{pk}(\mathrm{ID}(t'_2))$ in the second iteration, and so on. Here t'_s denotes the s^{th} nearest neighbor to Q, for $1 \leq s \leq k$. At the end of k iterations, only C_1 knows $\langle E_{pk}(\mathrm{ID}(t'_1)), \ldots, E_{pk}(\mathrm{ID}(t'_k))\rangle$. To start with, in the first iteration, C_1 and C_2 jointly compute the encryptions of the individual bits of the minimum value among $d_1, \ldots, d_n$ using SMIN$_n$. That is, C_1 with input $\langle [d_1], \ldots, [d_n]\rangle$ and C_2 compute $[d_{\min}]$, where $d_{\min}$ is the minimum value among $d_1, \ldots, d_n$. The output $[d_{\min}]$ is known only to C_1. Now, C_1 performs the following operations locally:

- Compute the encryption of $d_{\min}$ from its encrypted individual bits as below

$$\begin{aligned} E_{pk}(d_{\min}) &= \prod_{\gamma=0}^{l-1} E_{pk}(d_{\min,\gamma+1})^{2^{l-\gamma-1}} \\ &= E_{pk}(d_{\min,1} * 2^{l-1} + \cdots + d_{\min,l}) \end{aligned}$$

where $d_{\min,1}$ and $d_{\min,l}$ are the most and least significant bits of $d_{\min}$ respectively.

Algorithm 14 $\mathrm{S}k\mathrm{NN}_\mathrm{m}(E_{pk}(T), E_{pk}(Q)) \rightarrow \langle \mathrm{ID}(t'_1), \ldots, \mathrm{ID}(t'_k) \rangle$

Require: C_1 has $(E_{pk}(T), E_{pk}(Q))$ and π; C_2 has sk

1: C_1 and C_2: $E_{pk}(d_i) \leftarrow \mathrm{SSED}(E_{pk}(Q), E_{pk}(t_i))$ and $[d_i] \leftarrow \mathrm{SBD}(E_{pk}(d_i))$, for $1 \leq i \leq n$

2: **for** $s = 1$ to k **do:**

(a). C_1 and C_2: $[d_{\min}] \leftarrow \mathrm{SMIN}_n([d_1], \ldots, [d_n])$

(b). C_1:

- $E_{pk}(d_{\min}) \leftarrow \prod_{\gamma=0}^{l-1} E_{pk}(d_{\min,\gamma+1})^{2^{l-\gamma-1}}$
- **if** $s \neq 1$ **then**, for $1 \leq i \leq n$
  - $E_{pk}(d_i) \leftarrow \prod_{\gamma=0}^{l-1} E_{pk}(d_{i,\gamma+1})^{2^{l-\gamma-1}}$
- **for** $i = 1$ to n **do:**
  - $\tau_i \leftarrow E_{pk}(d_{\min}) * E_{pk}(d_i)^{N-1}$
  - $\tau'_i \leftarrow \tau_i^{r_i}$, where $r_i \in_R \mathbb{Z}_N$
- $\beta \leftarrow \pi(\tau')$; send β to C_2

(c). C_2:

- $\beta'_i \leftarrow D_{sk}(\beta_i)$, for $1 \leq i \leq n$
- Compute U, for $1 \leq i \leq n$:
  - **if** $\beta'_i = 0$ **then** $U_i = E_{pk}(1)$
  - **else** $U_i = E_{pk}(0)$
- Send U to C_1

(d). C_1:

- $V \leftarrow \pi^{-1}(U)$ and $V'_i \leftarrow V_i^{\mathrm{ID}(t_i)}$, for $1 \leq i \leq n$
- $E_{pk}(\mathrm{ID}(t'_s)) \leftarrow \prod_{i=1}^{n} V'_i$
- $\gamma_s \leftarrow E_{pk}(\mathrm{ID}(t'_s)) * E_{pk}(r_s)$, where $r_s \in_R \mathbb{Z}_N$
- Send γ_s to C_2 and r_s to Bob

(e). C_1 and C_2, for $1 \leq i \leq n$:

- $E_{pk}(d_{i,\gamma}) \leftarrow \mathrm{SBOR}(V_i, E_{pk}(d_{i,\gamma}))$, for $1 \leq \gamma \leq l$

3: C_2:

(a). **for** $1 \leq s \leq k$ **do**:

- $\gamma'_s \leftarrow D_{sk}(\gamma_s)$; send γ'_s to Bob

4: Bob:

(a). **for** $1 \leq s \leq k$ **do**:

- Receive r_s from C_1 and γ'_s from C_2
- $\mathrm{ID}(t'_s) \leftarrow \gamma'_s - r_s \mod N$

- Compute the encryption of difference between $d_{\min}$ and each d_i. That is, C_1 computes $\tau_i = E_{pk}(d_{\min}) * E_{pk}(d_i)^{N-1} = E_{pk}(d_{\min} - d_i)$, for $1 \leq i \leq n$.

- Randomize τ_i to get $\tau'_i = \tau_i^{r_i} = E_{pk}(r_i * (d_{\min} - d_i))$, where r_i is a random number in $\mathbb{Z}_N$. Note that τ'_i is an encryption of either 0 or a random number, for $1 \leq i \leq n$. Also, permute τ' using a random permutation function π (known only to C_1) to get $\beta = \pi(\tau')$ and send it to C_2.

Upon receiving β, C_2 decrypts it component-wise to get $\beta'_i = D_{sk}(\beta_i)$, for $1 \leq i \leq n$. After this, he/she computes an encrypted vector U of length n such that $U_i = E_{pk}(1)$ if $\beta'_i = 0$, and $E_{pk}(0)$ otherwise. This work assumes that exactly one of the entries in β equals to zero and rest of them are random. This further implies that exactly one of the entries in U is an encryption of 1 and rest of them are encryptions of 0's. However, if β' has more than one 0's, then C_2 can randomly pick one of those indexes and assign $E_{pk}(1)$ to the corresponding index of U and $E_{pk}(0)$ to the rest. Then, C_2 sends U to C_1. After receiving U, C_1 performs inverse permutation on it to get $V = \pi^{-1}(U)$. Note that exactly one of the entry in V is $E_{pk}(1)$ and the remaining are encryption of 0's. In addition, if $V_i = E_{pk}(1)$, then t_i is the closest profile vector to Q. However, C_1 and C_2 do not know which entry in V corresponds to $E_{pk}(1)$.

Now C_1 computes $E_{pk}(\mathrm{ID}(t'_1))$ locally, the encryption of the user ID of the closest profile to Q, and updates the distance vectors as follows:

- Compute $V'_i \leftarrow V_i^{\mathrm{ID}(t_i)}$, for $1 \leq i \leq n$. After this, by using homomorphic properties, C_1 computes the encrypted user ID of t_1 as $E_{pk}(\mathrm{ID}(t'_1)) = \prod_{i=1}^{n} V'_i$.

- It is important to note that the first nearest profile to Q should be obliviously excluded from further computations. However, since C_1 does not know the profile corresponding to $E_{pk}(\mathrm{ID}(t'_1))$, one need to obliviously eliminate the possibility of choosing this profile again in next iterations. For this, C_1 obliviously updates the distance corresponding to $E_{pk}(t'_1)$ to the maximum value, i.e., $2^l - 1$.

More specifically, C_1 updates the distance vectors with the help of C_2 using the SBOR protocol as below, for $1 \leq i \leq n$ and $1 \leq \gamma \leq l$.

$$E_{pk}(d_{i,\gamma}) = \text{SBOR}(V_i, E_{pk}(d_{i,\gamma}))$$

Note that when $V_i = E_{pk}(1)$, the corresponding distance vector d_i is set to the maximum value. That is, under this case, $[d_i] = \langle E_{pk}(1), \ldots, E_{pk}(1)\rangle$. However, when $V_i = E_{pk}(0)$, the OR operation has no affect on d_i.

The above process is repeated until k iterations, and in each iteration $[d_i]$ corresponding to the current chosen profile is set to the maximum value. However, since C_1 does not know which $[d_i]$ is updated, he/she has to re-compute $E_{pk}(d_i)$ in each iteration using the corresponding $[d_i]$, for $1 \leq i \leq n$. In iteration s, $E_{pk}(\text{ID}(t'_s))$ is known only to C_1.

At the end of the iterative step (i.e., step 2 of Algorithm 14), only C_1 has $\langle E_{pk}(\text{ID}(t'_1)), \ldots, E_{pk}(\text{ID}(t'_k))\rangle$ - the list of encrypted user IDs of k-nearest neighbors to the input query profile Q. Then C_1 proceeds as follows:

- Randomize the encrypted user IDs. More specifically, C_1 computes $E_{pk}(\gamma_s) = E_{pk}(\text{ID}(t'_s)) * E_{pk}(r_s)$, for $1 \leq s \leq k$. Here r_s is a random number in $\mathbb{Z}_N$ known only to C_1. Send γ_s to C_2 and r_s to Bob, for $1 \leq s \leq k$.

Upon receiving γ_s, C_2 decrypts it to get $\gamma'_s = D_{sk}(\gamma_s)$ and sends it to Bob, for $1 \leq s \leq k$. Note that, due to randomization by C_1, decryption operation on γ_s always yields a random number in $\mathbb{Z}_N$.

Finally, upon receiving r_s from C_1 and γ'_s from C_2, Bob computes the user ID of s^{th} potential friend as $\text{ID}(t'_s) = \gamma'_s - r_s \mod N$, for $1 \leq s \leq k$.

6.6.3. Security Analysis. First, due to the encryption of profile vectors and by semantic security of the Paillier cryptosystem, user's profile data is protected from C_1 and C_2 in both protocols.

In the SkNN$_{\rm b}$ protocol, the decryption operations at step 2(b) of Algorithm 13 reveal d_i values to C_2. In addition, since C_2 generates the top k index list (at step 2(c) of Algorithm 13) and sends it to C_1, the data access patterns are revealed to C_1 and C_2. In addition, the top k user IDs are revealed to C_1. Therefore, the basic SkNN$_{\rm b}$ protocol is secure under the assumption that d_i values can be revealed to C_2, output can be revealed to C_1, and data access patterns can be revealed to C_1 and C_2.

On the other hand, the security analysis of SkNN$_{\rm m}$ is as follows. At step 1 of Algorithm 14, the outputs of SSED and SBD are in encrypted format, and are known only to C_1. In addition, all the intermediate results decrypted by C_2 in SSED are uniformly random in $\mathbb{Z}_N$. Also, as mentioned in [100], the SBD protocol is secure. Thus, no information is revealed during step 1 of Algorithm 14. In each iteration, the output of SMIN$_n$ is known only to C_1 and no information is revealed to C_2. Also, C_1 and C_2 do not know which profile belongs to current global minimum. Thus, data access patterns are protected from both C_1 and C_2. At step 2(c) of Algorithm 14, a component-wise decryption of β reveals the tuples that satisfy the current global minimum distance to C_2. However, due to permutation by C_1, C_2 cannot trace back to the corresponding data profiles. Also, note that decryption of β gives either encryptions of 0's or random numbers in $\mathbb{Z}_N$. Similarly, since U is an encrypted vector, C_1 cannot know which tuple corresponds to current global minimum distance. Thus, data access patterns are further protected at this step from C_1. In addition, the update process at step 2(e) of Algorithm 14 does not leak any information to C_1 and C_2. In summary, C_1 and C_2 do not know which data profiles correspond to the output set $\langle t'_1, \ldots, t'_k \rangle$. Also, unlike S$kNN_{\rm b}$, it is worth pointing out that the output of SkNN$_{\rm m}$ is revealed only to Bob.

Based on the above discussions, it is clear that the proposed SkNN$_{\rm m}$ protocol protects the confidentiality of the data and hides the data access patterns from both C_1 and C_2.

6.6.4. Complexity Analysis. The computation complexity of SkNN$_{\mathrm{b}}$ is bounded by $O(n * m + k)$ encryptions, decryptions and exponentiations. In practice $k \ll n * m$; therefore, the computation complexity of SkNN$_{\mathrm{b}}$ is bounded by $O(n*m)$ encryptions and exponentiations (assuming that encryption and decryption operations under Paillier cryptosystem take similar amount of time).

On the other hand, the computation complexity of SkNN$_{\mathrm{m}}$ is bounded by $O(n)$ instantiations of SBD and SSED, $O(k)$ instantiations of SMIN$_n$, and $O(n*k*l)$ instantiations of SBOR. Note that the computation complexity of the SBD protocol proposed in [100] is bounded by $O(l)$ encryptions and $O(l)$ exponentiations. Also, the computation complexity of SSED is bounded by $O(m)$ encryptions and $O(m)$ exponentiations. In addition, the computation complexity of SMIN$_n$ is bounded by $O(l*n*\log_2 n)$ encryptions and $O(l*n*\log_2 n)$ exponentiations. Since SBOR utilizes SM as a sub-routine, the computation cost of SBOR is bounded by (small) constant number of encryptions and exponentiations. Based on the above analysis, the total computation complexity of the SkNN$_{\mathrm{m}}$ protocol is bounded by $O(n * (l + m + k * l * \log_2 n))$ encryptions and exponentiations.

6.7. EMPIRICAL ANALYSIS

This sub-section discusses the performances of the proposed protocols in detail under different parameter settings. By using Paillier cryptosystem [55], the proposed protocols were implemented in C. Various experiments were conducted on a Linux machine with an Intel® Xeon® Six-Core™ CPU 3.07 GHz processor and 12GB RAM running Ubuntu 10.04 LTS.

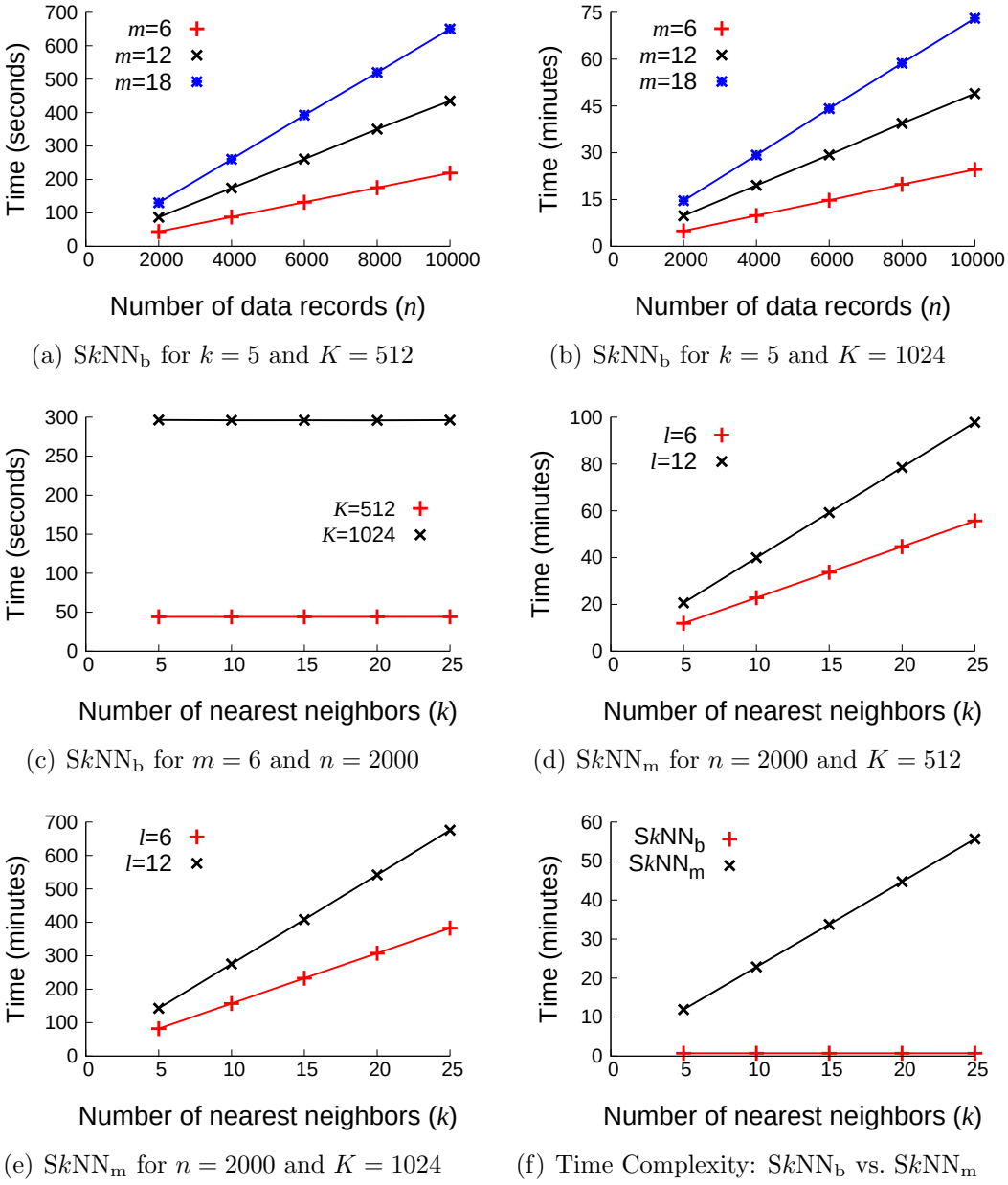
Since it is difficult to control the parameters in a real dataset, synthetic datasets are randomly generated depending on the parameter values in consideration. Using these synthetic datasets one can perform a more elaborated analysis on

the computation costs of the proposed protocols under different parameter settings. The datasets were encrypted attribute-wise, using the Paillier encryption whose key size is varied in the experiments, and the encrypted data were stored on the above machine. Then, a random query was chosen and executed over the encrypted data based on the protocols protocols. For the rest of this section, the performance of Bob is not discussed as he will not participate in any computations. Instead, the evaluations are based on the performance of federated cloud in SkNN$_{\mathrm{b}}$ and SkNN$_{\mathrm{m}}$ separately. In addition, the computation costs of the two protocols are compared. In the experiments, the Paillier encryption key size K is set to either 512 or 1024 bits.

6.7.1. Performance of SkNN$_{\mathbf{b}}$. This sub-section analyzes the computation costs of SkNN$_{\mathrm{b}}$ by varying the number of profiles (n), number of attributes in a profile (m), number of nearest neighbors (k), and encryption key size (K). The results are as shown in Figure 6.2. Note that the SkNN$_{\mathrm{b}}$ protocol is independent of the domain size of attributes (l).

First, by fixing $k = 5$ and $K = 512$, the computation costs of SkNN$_{\mathrm{b}}$ are evaluated for varying n and m. As shown in Figure 6.2(a), the computation costs of SkNN$_{\mathrm{b}}$ grows linearly with n and m. For example, when $m = 6$, the computation time of SkNN$_{\mathrm{b}}$ increases from 44.08 to 87.91 seconds when n is varied from 2000 to 4000. A similar trend is observed for $K = 1024$ as shown in Figure 6.2(b). For any fixed parameters, the observation is that the computation time of SkNN$_{\mathrm{b}}$ increases almost by a factor of 7 when K is doubled.

Next, by fixing $m = 6$ and $n = 2000$, the running times of SkNN$_{\mathrm{b}}$ for varying k and K are computed. The results are shown in Figure 6.2(c). Irrespective of K, the computation time of SkNN$_{\mathrm{b}}$ does not change much with varying k. This is because most of the cost in SkNN$_{\mathrm{b}}$ comes from the SSED protocol which is independent of k. E.g., when $K = 512$ bits, the computation time of SkNN$_{\mathrm{b}}$ changes from 44.08 to 44.14 seconds when k is changed from 5 to 25. Based on the above discussions, it is



(a) $Sk\mathrm{NN_b}$ for $k = 5$ and $K = 512$

(b) $Sk\mathrm{NN_b}$ for $k = 5$ and $K = 1024$

(c) $Sk\mathrm{NN_b}$ for $m = 6$ and $n = 2000$

(d) $Sk\mathrm{NN_m}$ for $n = 2000$ and $K = 512$

(e) $Sk\mathrm{NN_m}$ for $n = 2000$ and $K = 1024$

(f) Time Complexity: $Sk\mathrm{NN_b}$ vs. $Sk\mathrm{NN_m}$

Figure 6.2: Time complexities of $Sk\mathrm{NN_b}$ and $Sk\mathrm{NN_m}$ for varying values of n, m, l, k and encryption key size K

clear that the running time of $Sk\mathrm{NN_b}$ mainly depends on (or grows linearly with) n and m which further justifies the complexity analysis in Section 6.6.4.

6.7.2. Performance of $Sk\mathrm{NN_m}$. The computation costs of $Sk\mathrm{NN_m}$ for varying values of k, l and K are evaluated. Throughout this sub-section, the values

of m and n are fixed to 6 and 2000, respectively. However, the running time of SkNN$_{\mathrm{m}}$ grows almost linearly with n and m.

For $K = 512$ bits, the computation costs of SkNN$_{\mathrm{m}}$ for varying k and l are as shown in Figure 6.2(d). Following from Figure 6.2(d), for $l = 6$, the running time of SkNN$_{\mathrm{m}}$ varies from 11.93 to 55.65 minutes when k is changed from 5 to 25 respectively. Also, for $l = 12$, the running time of SkNN$_{\mathrm{m}}$ varies from 20.68 to 97.8 minutes when k is changed from 5 to 25 respectively. In either case, the cost of SkNN$_{\mathrm{m}}$ grows almost linearly with k and l.

A similar trend is observed for $K = 1024$ as shown in Figure 6.2(e). In particular, for any given fixed parameters, the computation cost of SkNN$_{\mathrm{m}}$ increases by almost a factor of 7 when K is doubled. For example, when $k = 10$, SkNN$_{\mathrm{m}}$ took 22.85 and 157.17 minutes to generate the 10 nearest neighbors of Q under $K = 512$ and 1024 bits respectively. Furthermore, when $k = 5$, the observation is that around 69.7% of cost in SkNN$_{\mathrm{m}}$ is accounted due to SMIN$_n$ which is initiated k times in SkNN$_{\mathrm{m}}$ (once in each iteration). Also, the cost incurred due to SMIN$_n$ increases from 69.7% to at least 75% when k is increased from 5 to 25.

In addition, by fixing $n = 2000, m = 6, l = 6$ and $K = 512$, the running times of both protocols are compared for varying values of k. As shown in Figure 6.2(f), the running time of SkNN$_{\mathrm{b}}$ remains to be constant at 0.73 minutes since it is almost independent of k. However, the running time of SkNN$_{\mathrm{m}}$ changes from 11.93 to 55.65 minutes as the value of k increases from 5 to 25.

Based on the above results, it is clear that the computation costs of SkNN$_{\mathrm{m}}$ are significantly higher than that of SkNN$_{\mathrm{b}}$. However, SkNN$_{\mathrm{m}}$ is more secure than SkNN$_{\mathrm{b}}$; therefore, the two protocols act as a trade-off between security and efficiency. Also, it is important to note that user's computation cost is mainly due to the encryption of his/her profile vector during outsourcing. As an example, for $m = 6$, Bob's computation costs are 4 and 17 milliseconds when K is 512 and 1024

bits respectively. In the proposed protocols, it is worth pointing out that users do not involve in any computations; therefore, they are very efficient from the end-user's perspective.

6.7.3. Towards Performance Improvement. At first, it seems that the proposed protocols are costly and may not scale well for large values of n. However, in both protocols, the computations involved on each profile are independent of others. Therefore, one can parallelize the operations on profile vectors for efficiency purpose. To further justify this claim, this work implemented a parallel version of the SkNN$_{\mathrm{b}}$ protocol using OpenMP programming and compared its computation costs with its serial version. As mentioned earlier, the machine used in this experiments has 6 cores which can be used to perform parallel operations on 6 threads. For $m = 6, k = 5$ and $K = 512$ bits, the comparison results are shown in Figure 6.3. The observation is that the parallel version of SkNN$_{\mathrm{b}}$ is roughly 6 times more efficient than its serial version. This is because of the fact that the parallel version can execute operations on 6 data records at a time (i.e., on 6 threads in parallel). E.g., when $n = 10000$, the running times of parallel and serial versions of SkNN$_{\mathrm{b}}$ are 40 and 215.59 seconds respectively.

Similar efficiency gains can be achieved by parallelizing the operations in SkNN$_{\mathrm{m}}$. Based on the above discussions, especially in a cloud computing environment where high performance parallel processing can be easily achieved, this work claims that the scalability issue of the proposed protocols can be eliminated or mitigated. In addition, using the existing map-reduce techniques, one can drastically improve the performance further by executing parallel operations on multiple nodes.

Following from the above empirical analysis, it is clear that SMIN$_n$ is the most costly sub-routine utilized in SkNN$_{\mathrm{m}}$. Therefore, by improving the efficiency of SMIN$_n$, one can improve the overall computation cost of SkNN$_{\mathrm{m}}$.

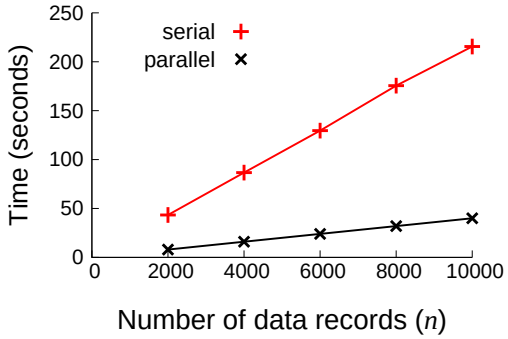


Figure 6.3: Parallel vs. serial versions of SkNN$_{\mathrm{b}}$ for $m = 6, k = 5$ and $K = 512$

6.8. CONCLUSION

k-nearest neighbors is one of the commonly used query in many data mining applications such as detection of fraud by credit card companies and prediction of tumor cells levels in blood. With the recent growth of cloud computing as a new IT paradigm, data owners are more interested to outsource their databases as well as DBMS functionalities to the cloud. Under an outsourced social networking environment, where encrypted users' profile data are stored in the cloud, secure query processing over encrypted data becomes challenging. In the literature, various secure k-nearest neighbor (SkNN) techniques have been proposed. However, the existing SkNN techniques over encrypted data are not secure.

Along this direction, two novel SkNN protocols over encrypted data in the cloud are proposed. The first protocol, which acts as a basic solution, leaks some information to the cloud. On the other hand, the second protocol is fully secure, that is, it protects the confidentiality of the data and also hides the data access patterns. However, the second protocol is more expensive compared to the basic protocol. Also, this work evaluated the performance of both protocols under different parameter settings. As a future work, it will be interesting to investigate and extend this research to other complex conjuctive queries over encrypted data.

7. CONCLUSIONS AND POSSIBLE FUTURE RESEARCH

The popularity of online social networks (OSNs) is on a constant rise due to various advantages such as online communication and sharing information of interest among friends. In general, users often wish to make new friends; therefore, improving the chances of expanding their social connections as well as getting information from a broader range of friends. Friend recommendation is a well-known application in many OSNs and has been studied extensively in the recent past. However, with the growing concerns about user privacy, there is a strong need to develop privacy-preserving friend recommendation methods for social networks. Therefore, this document proposed a set of private friend recommendation protocols as a way to facilitate the friend recommendation process possible even when the user's information in consideration is kept as private.

The proposed protocol in Section 3 computes the recommendation scores of all users within a radius of h from the target user A by using the similarity metric proposed in [14] as a baseline. More specifically, the proposed protocol generates the (scaled) recommendation scores along with the corresponding user IDs in such a way that the relative ordering among the users in the TOP-K list of recommended users is preserved (i.e., same accuracy as in [14]). In addition, this work demonstrated a new security issue in the current online social networks due to the inherent message flow information between different entities. To mitigate this issue or to provide better security, this work also proposed an extended version of the proposed protocol using randomization technique. Furthermore, a detailed empirical analysis based on different parameter settings were provided.

The proposed protocols in Section 4 facilitate users in a group G to get friend recommendations based on the social network structure and the users' social tags. For

a target user u_i, the proposed protocols compute the social closeness scores between u_i and each user in the subset $G_i \subset G$ in a privacy-preserving manner by utilizing an ontology tree T constructed by the domain expert such as the network administrator.

The proposed protocols in Section 5 recommend new friends to a given target user A using the common neighbors proximity measure under the assumption that users' friend lists are private. The first protocol PPFR_h is based on an additive homomorphic encryption scheme, and its accuracy is essentially based on the parameters of universal hash function $h_{a,b}$. Whereas the second protocol PPFR_{sp} utilizes the concept of protecting the source privacy through randomizing the message passing path and also recommends friends accurately. The PPFR_{sp} protocol is more efficient than the PPFR_h protocol. The proposed PPFR protocols act as a trade-off among security, efficiency and accuracy.

The proposed protocols in Section 6 generate friend recommendations to a given target user assuming an outsourced social networking environment, where users' profile data are encrypted and stored in the cloud. Both protocols are constructed based on the k-nearest neighbor technique. That is, the proposed protocols compute k-nearest profiles (in encrypted form) to a given target profile and obliviously recommend the corresponding candidates as top k potential friends to the target user. The experimental results showed that the $\text{S}k\text{NN}_\text{b}$ protocol is significantly more efficient than $\text{S}k\text{NN}_\text{m}$. However, $\text{S}k\text{NN}_\text{m}$ provides better security than $\text{S}k\text{NN}_\text{b}$.

The proposed protocols in Sections 3 to 5 assume that either users' friend lists, social tags, or messages exchanged with other users of an online social network (OSN) as private information. As a future work, it is also desirable to develop protocols that can recommend friends based on other details such as education and employment in a privacy-preserving manner. Also, the proposed protocols in Section 4 assume that the network administrator builds the ontology tree T based on the domain knowledge of a particular group. However, extending it to multiple groups may not seem to be

feasible at this point of time; therefore, it can be treated as an interesting direction for future research.

In the literature, many friend recommendation protocols have been proposed based on different similarity metrics. However, only recently, researchers have focused on constructing hybrid friend recommendation protocols by taking both network structure as well as users' profile contents into consideration. Hence, a possible extension to this work is to explore alternative ways for developing hybrid privacy-preserving friend recommendation methods by combining different scoring functions.

BIBLIOGRAPHY

[1] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[2] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 611–617, 2006.

[3] D. Boyd and N. B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2008.

[4] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42, 2007.

[5] D. M. Ehrlich. Social network survey paper. *International Journal of Learning and Intellectual capital*, 2006.

[6] F. Bonchi, C. Castillo, A. Gionis, and A. Jaimes. Social network analysis and mining for business applications. *ACM Transactions on Intelligent Systems and Technology*, 2:22:1–22:37, May 2011.

[7] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 135–142, 2010.

[8] P. Symeonidis, E. Tiakas, and Y. Manolopoulos. Product recommendation and rating prediction based on multi-modal social networks. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 61–68, 2011.

[9] J. Chen, W. Geyer, C. Dugan, M. Muller, and I. Guy. Make new friends, but keep the old: recommending people on social networking sites. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 201–210, 2009.

[10] J. Naruchitparames, M. H. Giine, and S. J. Louis. Friend recommendations in social networks using genetic algorithms and network topology. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2207 –2214, 2011.

[11] N. B. Silva, I. R. Tsang, G. D. C. Cavalcanti, and I. J. Tsang. A graph-based friend recommendation system using genetic algorithm. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1 –7, July 2010.

[12] X. Xie. Potential friend recommendation in online social network. In *IEEE/ACM Int'l Conference on Cyber, Physical and Social Computing and Int'l Conference on Green Computing and Communications*, pages 831 –835, 2010.

[13] L. Gou, F. You, J. Guo, L. Wu, and X. Zhang. Sfviz: interest-based friends exploration and recommendation in social networks. In *Proceedings of Visual Information Communication - International Symposium*, VINCI '11, pages 1–10. ACM, 2011.

[14] B. Dai, C. Lee, and C. Chung. A framework of recommendation system based on both network structure and messages. In *International Conference on Advances in Social Networks Analysis and Mining (ASONAM' 11)*, pages 709 –714, July 2011.

[15] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of American Society for Information Science and Technology*, 58:1019–1031, 2007.

[16] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 556–559, New York, 2003. ACM.

[17] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, WPES '05, pages 71–80, 2005.

[18] C. Dwyer, S. R. Hiltz, and K. Passerini. Trust and privacy concern within social networking sites: A comparison of facebook and myspace. In *Proceedings of the Thirteenth Americas Conference on Information systems (AMCIS 2007)*, 2007.

[19] B. Krishnamurthy and C. E. Wills. Characterizing privacy in online social networks. In *Proceedings of the first workshop on Online social networks*, WOSN '08, pages 37–42. ACM, 2008.

[20] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu. Link privacy in social networks. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 289–298. ACM, 2008.

[21] J. Delgado, E. Rodríguez, and S. Llorente. User's privacy in applications provided through social networks. In *Proceedings of second ACM SIGMM workshop on Social media*, WSM '10, pages 39–44, 2010.

[22] H. Gao, J. Hu, T. Huang, J. Wang, and Y. Chen. Security issues in online social networks. *IEEE Internet Computing*, 15(4):56 –63, 2011.

[23] L. A. Cutillo, R. Molva, and M. Onen. Analysis of privacy in online social networks from the graph theory perspective. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 2011)*, pages 1 –5, Dec. 2011.

[24] M. Huber, M. Mulazzani, S. Schrittwieser, and E. Weippl. Cheap and automated socio-technical attacks based on social networking sites. In *Proceedings of the 3rd ACM workshop on Artificial intelligence and security*, AISec '10, pages 61–64. ACM, 2010.

[25] D. Irani, M. Balduzzi, D. Balzarotti, E. Kirda, and C. Pu. Reverse social engineering attacks in online social networks. In *Proceedings of the 8th international conference on Detection of intrusions and malware, and vulnerability assessment*, DIMVA'11, pages 55–74, Berlin, Heidelberg, 2011. Springer-Verlag.

[26] R. Dey, Z. Jelveh, and K. Ross. Facebook users have become much more private: A large-scale study. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 346 –352, march 2012.

[27] G. Swamynathan, C. Wilson, B. Boe, K. Almeroth, and B. Y. Zhao. Do social networks improve e-commerce?: a study on social marketplaces. In *Proceedings of the first workshop on Online social networks*, WOSN '08, pages 1–6, New York, 2008. ACM.

[28] S. Asur and B. A. Huberman. Predicting the future with social media. In *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 492–499, 2010.

[29] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5:1–5:39, May 2007.

[30] S. Zhao, M. X. Zhou, X. Zhang, Q. Yuan, W. Zheng, and R. Fu. Who is doing what and when: Social map-based recommendation for content-centric social web sites. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(1):5:1–5:23, October 2011.

[31] J. He and W. W. Chu. A social network-based recommender system (snrs). *Annuals of Information Systems, Special issue on Data Mining for Social Network Data*, 12:47–74, 2010.

[32] H. Ma, T. C. Zhou, M. R. Lyu, and I. King. Improving recommender systems by incorporating social contextual information. *ACM Trans. Inf. Syst.*, 29:1–23, April 2011.

[33] L. M. Aiello, A. Barrat, R. Schifanella, C. Cattuto, B. Markines, and F. Menczer. Friendship prediction and homophily in social media. *ACM Transactions on the Web (TWEB)*, 6(2):9:1–9:33, June 2012.

[34] F. Ratiu. Facebook blog - people you may know, 5 December 2010. `http://blog.facebook.com/blog.php?post=15610312130`.

[35] W. Dong, V. Dave, L. Qiu, and Y. Zhang. Secure friend discovery in mobile social networks. In *Proceedings IEEE INFOCOM*, pages 1647 –1655, April 2011.

[36] A. Machanavajjhala, A. Korolova, and A. D. Sarma. Personalized social recommendations: accurate or private. *Proc. VLDB Endowment*, 4:440–450, april 2011.

[37] C. Dwork. Differential privacy. In *the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1–12. Springer, 2006.

[38] A. C. Yao. Protocols for secure computation. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 160–164. IEEE, 1982.

[39] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167. IEEE, 1986.

[40] M. R. Clarkson, S. Chong, and A.C. Myers. Civitas: Toward a secure voting system. In *IEEE Symposium on Security and Privacy*, pages 354 –368, May 2008.

[41] C. Cachin. Efficient private bidding and auctions with an oblivious third party. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 120–127, Kent Ridge Digital Labs, Singapore, 1999. ACM Press.

[42] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*. ACM Press, 1999.

[43] R. Agrawal and R. Srikant. Privacy preserving data mining. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD International Conference on Management of data*, volume 29, pages 439–450, 2000.

[44] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Journal of Cryptology*, volume 15, pages 177 – 206, 2002.

[45] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. In *19th ACM Symposium on the Theory of Computing*, pages 218–229, New York, New York, United States, 1987.

[46] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[47] A. Ben-David, N. Nisan, and B. Pinkas. Fairplaymp - a system for secure multiparty computation. In *Proceedings of the ACM Computer and Communications Security Conference (ACM CCS)*, October 2008.

[48] J. Katz and Y. Lindell. *Introduction to Modern Cryptography.* CRC Press, 2007.

[49] O. Goldreich. *The Foundations of Cryptography*, volume 2, chapter Encryption Schemes, pages 373–470. Cambridge, University Press, 2004.

[50] O. Goldreich. *The Foundations of Cryptography*, volume 2, chapter General Cryptographic Protocols. Cambridge, University Press, 2004.

[51] Y. Yang, J. Lutes, F. Li, B. Luo, and P. Liu. Stalking online: on user privacy in social networks. In *Proceedings of the second ACM conference on Data and Application Security and Privacy (CODASPY '12)*, pages 37–48. ACM, 2012.

[52] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[53] S. Lo and C. Lin. Wmr–a graph-based algorithm for friend recommendation. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI '06)*, pages 121–128. IEEE Computer Society, 2006.

[54] C. Dwork. Differential privacy: a survey of results. In *Proceedings of the 5th international conference on Theory and applications of models of computation*, TAMC'08, pages 1–19, Berlin, Heidelberg, 2008. Springer-Verlag.

[55] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th international conference on Theory and application of cryptographic techniques.* Springer-Verlag, 1999.

[56] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, 18:186–208, February 1989.

[57] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: an online social network with user-defined privacy. *ACM SIGCOMM Computer Communication Review*, 39(4):135–146, August 2009.

[58] NIST. Advanced encryption standard. Technical Report NIST Special Publication FIPS-197, National Institute of Standards and Technology, 2001.

[59] F. M. Suchanek, M. Vojnovic, and D. Gunawardena. Social tags: meaning and suggestions. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 223–232. ACM, 2008.

[60] W. Jiang, M. Murugesan, C. Clifton, and L. Si. Similar document detection with limited information disclosure. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 735–743. IEEE Computer Society, 2008.

[61] M. Murugesan, W. Jiang, C. Clifton, L. Si, and J. Vaidya. Efficient privacy-preserving similar document detection. *The VLDB Journal*, 19:457–475, August 2010.

[62] L. Gou, S. Zhang, J. Wang, and X. Zhang. Tagnetlens: multiscale visualization of knowledge structures in social tags. In *Proceedings of the 3rd International Symposium on Visual Information Communication*, VINCI '10, pages 18:1–9. ACM, 2010.

[63] W. Jiang and C. Clifton. AC-framework for privacy-preserving collaboration. In *SIAM International Conference on Data Mining*, Minneapolis, Minnesota, April 26-28 2007.

[64] W. Jiang, C. Clifton, and M. Kantarcioglu. Transforming semi-honest protocols to ensure accountability. *Data and Knowledge Engineering*, 2008.

[65] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215, Washington, DC, August 24-27 2003.

[66] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Eurocrypt 2004*, Interlaken, Switzerland, May 2-6 2004. International Association for Cryptologic Research (IACR).

[67] L. Kissner and D. Song. Privacy-preserving set operations. In *Advances in Cryptology - CRYPTO 2005, LNCS*, pages 241–257. Springer, 2005.

[68] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen. On secure scalar product computation for privacy-preserving data mining. In Choonsik Park and Seongtaek Chee, editors, *The 7th Annual International Conference in Information Security and Cryptology (ICISC 2004)*, pages 104–120, Seoul, Korea, December 2-3 2004.

[69] W. Jiang and B. K. Samanthula. N-gram based secure similar document detection. Technical Report TR 2011-01, Department of Computer Science, Missouri S & T, Rolla, Missouri, February 2011. `http://web.mst.edu/~wjiang/ngram-ssdd-tech.pdf`.

[70] M. M. Lucas and N. Borisov. Flybynight: mitigating the privacy risks of social networking. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society*, WPES '08, pages 1–8, New York, NY, USA, 2008. ACM.

[71] S. Guha, K. Tang, and P. Francis. Noyb: privacy in online social networks. In *Proceedings of the first workshop on Online social networks*, WOSN '08, pages 49–54, New York, NY, USA, 2008. ACM.

[72] A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman. Lockr: better privacy for social networks. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 169–180, New York, NY, USA, 2009. ACM.

[73] J. Sun, X. Zhu, and Y. Fang. A privacy-preserving scheme for online social networks with efficient revocation. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 2516–2524, Piscataway, NJ, USA, 2010. IEEE Press.

[74] S. Jahid, P. Mittal, and N. Borisov. Easier: encryption-based access control in social networks with efficient revocation. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, pages 411–415, New York, NY, USA, 2011. ACM.

[75] E. D. Cristofaro, C. Soriente, G. Tsudik, and A. Williams. Hummingbird: Privacy at the time of twitter. In *IEEE Symposium on Security and Privacy*, pages 285–299, 2012.

[76] A. J. Feldman, A. Blankstein, M. J. Freedman, and E. W. Felten. Social networking with frientegrity: privacy and integrity with an untrusted provider. In *Proceedings of the 21st USENIX conference on Security symposium*, Security'12, pages 31–31, Berkeley, CA, USA, 2012. USENIX Association.

[77] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *Proceedings of the 2001 workshop on New security paradigms*, NSPW '01, pages 13–22. ACM, 2001.

[78] W. Jiang, L. Si, and J. Li. Protecting source privacy in federated search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 761–762, 2007.

[79] E. Steel and J. E. Vascellaro. Facebook, myspace confront privacy loophole, May 2010.

[80] J. P. Mello. Facebook scrambles to fix security hole exposing private pictures, 20 September 2012. `http://www.pcworld.com/article/245582/`.

[81] D. Grippi, M. Salzberg, R. Sofaer, and I. Zhitomirskiy. Diaspora project, 13 October 2012. `http://diasporaproject.org/`.

[82] L. A. Cutillo, R. Molva, and T. Strufe. Safebook: Feasibility of transitive cooperation for privacy on a decentralized social network. In *WOWMOM*, pages 1–6, 2009.

[83] D. Liu, A. Shakimov, R. Cáceres, A. Varshavsky, and L. P. Cox. Confidant: protecting osn data without locking it up. In *Proceedings of the 12th ACM/IFIP/USENIX international conference on Middleware*, Middleware'11, pages 61–80, Berlin, Heidelberg, 2011. Springer-Verlag.

[84] A. Shakimov, H. Lim, R. Caceres, L.P. Cox, K. Li, Dongtao Liu, and A. Varshavsky. Vis-à-vis: Privacy-preserving online social networking via virtual individual servers. In *Third International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–10. IEEE, 2011.

[85] S. Nilizadeh, N. Alam, N. Husted, and A. Kapadia. Pythia: a privacy aware, peer-to-peer network for social search. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, WPES '11, pages 43–48, New York, NY, USA, 2011. ACM.

[86] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, and A. Kapadia. Decent: A decentralized architecture for enforcing privacy in online social networks. In *PerCom Workshops*, pages 326–332. IEEE, 2012.

[87] S. Nilizadeh, S. Jahid, P. Mittal, N. Borisov, and A. Kapadia. Cachet: a decentralized architecture for privacy preserving social networking with caching. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, CoNEXT '12, pages 337–348, New York, NY, USA, 2012. ACM.

[88] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[89] B. K. Samanthula and W. Jiang. Structural and message based private friend recommendation. In *Proceedings of IEEE International Conference on Advances in Social Networks Analysis and Mining*, August 2012.

[90] K. Frikken. Privacy-preserving set union. In Jonathan Katz and Moti Yung, editors, *Applied Cryptography and Network Security*, volume 4521 of *Lecture Notes in Computer Science*, pages 237–252. Springer Berlin Heidelberg, 2007.

[91] L. Kissner, A. Oprea, M. K. Reiter, D. Song, and K. Yang. Private keyword-based push and pull with applications to anonymous communication (extended abstract). In *Applied Cryptography and Network Security*, 2004.

[92] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*, chapter Data Structures. The MIT Press, 3rd edition, 2009.

[93] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review Letters E*, 64, 2001.

[94] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[95] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[96] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38:690–728, 1991.

[97] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology – EUROCRYPT*, pages 280–299, 2001.

[98] I. Damgard, M. Jurik, and J. B. Nielsen. A generalization of Paillier's public-key system with applications to electronic voting, 2003.

[99] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[100] B. K. Samanthula and W. Jiang. An efficient and probabilistic secure bit-decomposition. In *Proceedings of the 8th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '13, pages 541–546, Hangzhou, China, May 8-10 2013.

[101] H. Hu, J. Xu, C. Ren, and B. Choi. Processing private queries over untrusted data cloud through privacy homomorphism. In *ICDE*, pages 601–612. IEEE, 2011.

[102] P. Mell and T. Grance. The nist definition of cloud computing (draft). *NIST special publication*, 800:145, 2011.

[103] M. Li, S. Yu, W. Lou, and Y. T. Hou. Toward privacy-assured cloud data services with flexible search functionalities. In *ICDCSW*, pages 466–470. IEEE, 2012.

[104] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *VLDB*, pages 720–731, 2004.

[105] E. Shi, J. Bethencourt, T. Chan, D. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy (SP'07)*, pages 350–364. IEEE, 2007.

[106] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu. Secure multidimensional range queries over outsourced data. *The VLDB Journal*, 21(3):333–358, 2012.

[107] H. Hacıgümüş, B. Iyer, and S. Mehrotra. Efficient execution of aggregation queries over encrypted relational databases. In *Database Systems for Advanced Applications*, pages 125–136. Springer, 2004.

[108] E. Mykletun and G. Tsudik. Aggregation queries in the database-as-a-service model. In *Data and Applications Security XX*, pages 89–103. Springer, 2006.

[109] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis. Secure knn computation on encrypted databases. In *SIGMOD*, pages 139–152, 2009.

[110] Y. Zhu, R. Xu, and T. Takagi. Secure k-nn computation on encrypted cloud data without sharing key with query users. In *Cloud Computing*, pages 55–60. ACM, 2013.

[111] B. Yao, F. Li, and X. Xiao. Secure nearest neighbor revisited. In *IEEE ICDE*, Brisbane, Australia, April 2013.

[112] S. Bugiel, S. Nürnberger, A. Sadeghi, and T. Schneider. Twin clouds: An architecture for secure cloud computing (extended abstract). In *Workshop on Cryptography and Security in Clouds*, March 2011.

[113] J. Wang, H. Ma, Q. Tang, J. Li, H. Zhu, S. Ma, and X. Chen. Efficient verifiable fuzzy keyword search over encrypted data in cloud computing. *Computer Science and Information Systems*, 10(2):667–684, 2013.

[114] J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. *Information Security*, pages 471–483, 2002.

[115] M. Shaneck, Y. Kim, and V. Kumar. Privacy preserving nearest neighbor search. *Machine Learning in Cyber Trust*, pages 247–276, 2009.

[116] Y. Qi and M. J Atallah. Efficient privacy-preserving k-nearest neighbor search. In *ICDCS*, pages 311–319. IEEE, 2008.

[117] J. Vaidya and C. Clifton. Privacy-preserving top-k queries. In *ICDE*, pages 545–546. IEEE, 2005.

[118] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. Tan. Private queries in location based services: anonymizers are not necessary. In *SIGMOD*, pages 121–132. ACM, 2008.

VITA

Bharath Kumar Samanthula was born in Nellore, a coastal and one of the 23 districts of Andhra Pradesh, India. He received his bachelors degree in Computer Science and Engineering from International Institute of Information Technology, Hyderabad in 2008.

Bharath joined the Computer Science department at the Missouri University of Science and Technology as a graduate student in August 2008. He worked as a Graduate Research Assistant under the guidance of Dr. Wei Jiang and earned his Ph.D. degree in December 2013. His research interests include Applied Cryptography, Personal Privacy and Data Security in Social Networks, Cloud Computing and Smart Grids. During his time at Missouri S&T, he served as the Vice President for IEEE Computer Society Student Branch from October 2011 to February 2013. Also, he served as the Computer Science Department Representative for the Council of Graduate Students from 2010 to 2011.