TRANSACTIONS ON DATA PRIVACY 8 (2015) 141–171

Privacy-Preserving and Efficient Friend Recommendation in Online Social Networks

Bharath K. Samanthula*, **Lei Cen***, **Wei Jiang****, **Luo Si***

*Department of Computer Science, Purdue University, 305 N University St, West Lafayette, IN 47907-2066, USA.

**Department of Computer Science, Missouri University of Science and Technology, 310 Computer Science Building, 500 W 15th St, Rolla, MO 65409-0350, USA.

E-mail: bsamanth@purdue.edu, lcen@cs.purdue.edu, wjiang@mst.edu, lsi@cs.purdue.edu

Abstract. The popularity of online social networks (OSNs) is on constant rise due to various advantages, including online communication and sharing information of interest among friends. It is often that users want to make new friends to expand their social connections as well as to obtain information from a broad range of people. Friend recommendation is a very important 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. In this paper, we propose two novel methods to recommend friends for a given 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 anonymous message routing and recommends friends accurately and efficiently. In addition, we empirically compare the efficiency and accuracy of the proposed protocols, and address the implementation details of the two methods in practice. The proposed protocols provide a trade-off among security, accuracy, and efficiency; thus, users or the network provider can choose between these two protocols depending on the underlying requirements.

Keywords. Online Social Network, Privacy, Friend Recommendation, Encryption.

1 Introduction

An OSN facilitates users to stay in touch with others 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 emerging growth of OSNs [7,23,47], such as Facebook, MySpace, Twitter, YouTube, Google+ and LinkedIn, has resulted in vast amount of social data containing personal and sensitive information about individual users. Social network analyses [6,57] involve mining the social data to understand user activities and to identify the relationships among various users. Social network analyses have boosted the research in developing various recommendation algorithms such as [41,74]. 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.

Friend recommendation algorithms in OSNs can be based on network topology, user contents or both. Topology-based algorithms [9, 58, 70] focus on the topological structures of social networks as how users are currently connected. These algorithms check whether the friends of existing friends of user A can be recommended as new friends to A using some similarity measures. In contrast, the content-based algorithms [35, 77] utilize the profile information of users, such as education and hobbies, to recommend new friends to A. In this paper, we restrict our discussion to topology based friend recommendations.

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. In contrast, 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 [49, 50]. Therefore, in this paper, we use the common neighbors as the similarity measure to compute the social closeness between two users (see Section 4.3 for more details).

With the growing use of OSNs, there have been various concerns about user privacy [13, 17, 22, 28, 36, 45, 46]. 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. This paper proposes two novel methods for solving a specific mining problem in social networks, namely 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. Besides protecting the information being shared, users may also want to hide the list of users with whom they typically share their sensitive information (e.g., photos and articles), for privacy reasons. Therefore, protecting the friendship information in OSNs is of central importance which would otherwise poses a great threat to the user's privacy through social engineering attacks [39,40].

In addition, we assume that the social network operators or administrators are not allowed to know the users' friend lists or other private personal profile information. We emphasize that this is a commonly made assumption in the related problem domains [3, 12, 25, 37, 39, 53, 72, 75] 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, we believe that 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. We think there will be a market for such

private social networks (see Section 2.2 for more details).

1.1 Problem Definition

Suppose we have a social network where the friendship between two users is considered 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. We term this problem as privacy-preserving friend recommendation (PPFR), and the goal of this paper is to develop PPFR protocols based on the common neighbors method [50]. 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 $1 \leq i \leq m$. We formally define a PPFR protocol as follows:

$$\text{PPFR}(A, Fr(A), Fr(B_1), \ldots, Fr(B_m), t) \rightarrow S \tag{1}$$

where t denotes a threshold value (more details are given in Section 5) and S denotes the list of recommended friends whose common neighbors scores (for any two users, it is defined as the number of common friends between them) with A are greater than or equal to t. In general, a PPFR protocol should satisfy the following requirements:

1. **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.
2. **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.

We emphasize that our 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 web-page 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).

In order to achieve the privacy requirements of PPFR mentioned above, we model the PPFR problem as a Secure Multiparty Computation (SMC) [29] problem due to the following reasons. In general, the SMC models are widely used in developing various privacy-preserving applications. This is because the main goal of building any privacy-preserving application is to let multiple parties to evaluate a common functionality without compromising their privacy. That is, apart from the information they already know (which typically consists of their private inputs), the participating parties should not deduce any additional information other than the outputs specified by the protocol. Under SMC, given n parties, each holding a private input a_i, the goal is to securely evaluate a common functionality f such that the output b_i is known only to party i. Depending on the application requirements, the value of b_i that would be known to party i can be varied. Also, during this process, except $< a_i, b_i >$, no other information is revealed to party i. Such privacy requirements are needed in most real-world applications, including the privacy-preserving friend recommendation (PPFR) problem which we justify below.

To develop mathematically strong (in terms of security guarantee) PPFR algorithms, we used the well-known security definitions and threat models in SMC (more details are given in Section 3). In the PPFR problem, the final output (i.e., the newly recommended friend list) should be revealed only to the target user A. That is, the PPFR problem can be formulated as follows: given a target user A (with his/her friend list as private input), the friends of A and/or the friends of A's friends (each holding their respective friend list as his/her private input), the goal of the PPFR is to compute the new friends for A using the common neighbors method (i.e., f) such that the final output is revealed only to A. Under this case, it is implicit that the output of all the other participating users (i.e., except A) is zero. By combining the above results, it is clear that the privacy requirements of the PPFR problem are the same as the privacy offered by the SMC model discussed above (see Sections 3 and 5 for more details).

1.2 Our Contributions

In this paper, we present two novel PPFR protocols that satisfy the well-known security definitions in the literature of Secure Multiparty Computation (SMC) [19,30]. The first protocol adopts an additive homomorphic encryption scheme [63] 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. On the other hand, the second method utilizes the concept of protecting the source privacy [42] through anonymous message routing and recommends friends accurately. The main contributions of this paper can be summarized as follows:

- **Security -** Both of the proposed protocols preserve the privacy of each user (i.e., his/her friend list). Here a user's friend list is protected from other users in the social network, the network administrator, and any other adversary as well. 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 the protocols is provided in Section 5.

- **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. More details are provided in Section 6.

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

The remainder of this paper is organized as follows: Section 2 summarizes the existing related work. Section 3 discusses the security model adopted in this paper. Section 4 presents some concepts and properties, as a background, that are used throughout this paper. Section 5 discusses the proposed protocols in detail along with a running example. We empirically compare the efficiency and accuracy of the proposed protocols by providing various experimental results in Section 6. Finally, we demonstrate the implementation details of the proposed protocols in Section 7 and conclude the paper with future work in Section 8.

2 Related Work

In this section, we first review upon the existing work related to friend recommendations in OSNs. Then we discuss about private social networks and the existing PPFR algorithms.

2.1 Existing Friend Recommendation Algorithms in OSNs

Social network analyses have been utilized for various business applications [6,73], such as predicting the future [2] and developing recommender systems [38, 48, 54, 80]. 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. [70] based on a genetic algorithm to optimize the three indices derived from the structural properties of social networks. Xie [77] 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. [58] developed a two-step approach to provide quality friend recommendations by combining cognitive theory with a Pareto-optimal genetic algorithm. Gou et al. [35] 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 was studied by Aiello et al. [1] 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 to users based on the simple "friend-of-a-friend" approach [24].

It is important to note that all the above methods are based on the assumption that the social network data are available to the network administrator in clear format. As such, they are not applicable to the PPFR problem.

2.2 Private Social Networks

In the 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 by the users). In addition, with the past history of data leaks and privacy controversies [56,71], users have many trust issues with OSN providers. To address such 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, significant work has been published (e.g., [14,40,51,60,64,68]).

Most recently, Nilizadeh et al. [61] 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 [5], 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 speed up the process of loading the data in newsfeed application. We emphasize that their scheme is entirely different from ours. First, their decentralized architecture involves no network provider and the communication happens directly between users. In our model, user's data are encrypted and stored on the server of an OSN. Therefore, our problem setting is orthogonal to their model. Second, the algorithm proposed in [61] is to support newsfeed application (under peer-to-peer network architecture) whereas this paper focuses on the friend recommendation application. Third, we emphasize that the peer-to-peer based social network system involves heavy computation as well as communication costs on the users. Thus, in this paper, we focus on conventional OSN framework where the data reside on the server of an OSN.

We emphasize that the decentralized architecture in OSNs is a more restricted model than the centralized architecture (which is a typical model in the current OSNs, e.g., Facebook and Google+) since it avoids the use of an OSN provider altogether. Also, it was claimed in [25] 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, our work considers a OSN system with a centralized provider. However, due to privacy and security concerns, we assume that user's can encrypt their profile data and store it on the OSN's server. Nevertheless, even after proper encryption of user's data, the 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 [25].

Along this direction, the authors in [25] 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 [25], Frientegrity reveals social relations to the service provider whereas in our 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 our work, friend list of a user is never revealed to other users.

2.2.1 Existing PPFR Algorithms in OSNs

Due to various privacy issues [13, 17, 22, 28, 36, 45, 46, 76], 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. [18] proposed a method to securely compute and verify social proximity between two users using cosine similarity in mobile social networks. Our work differs from theirs in two aspects. First, our problem setting is entirely different from theirs. We assume that users' friend lists are kept secret; thus, who are involved in the computation is not revealed before hand. In contrast, 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, our protocols 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 the user privacy.

Machanavajjhala et al. [55] estimated the trade-offs between accuracy and privacy of private friend recommendations using differential privacy [20, 21]. 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. However, differential privacy model has certain limitations and is mainly suitable for statistical databases. For example, it was mentioned in [55] that if privacy has to be preserved under the differential privacy model 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 contrast, our privacy guarantees are based on an entirely different security model, namely Secure Multiparty Computation (SMC) [30, 78, 79] model. As mentioned earlier, the SMC model is a widely used model in cryptography and it has stronger security properties compared to the differential privacy model. Under the SMC model, we are able to develop accurate PPFR protocols. In particular, our 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. [66] proposed a new private friend recommendation algorithm using a specific scoring function [15, 52] 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 ours. However, their method leaks valuable information such as whether two users (who are friends and the same number of hops away from A in the corresponding candidate network) share a common friend or not. This point was also mentioned in [66] as a drawback. Nevertheless, the scoring function adopted in our protocols (i.e., common neighbors method) is entirely different from theirs, and the protocols proposed in this paper are more secure in the perspective of protecting friendship information.

2.3 Secure Set Operations

At the first glance, secure set operation protocols [26, 27, 44] 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 us illustrate the challenges using the protocols proposed in [44] since these protocols seem most suitable to solve the proposed friend recommendation problem.

The secure set operation protocols given in [44] are based on representing a set (i.e., a friend list in our case) as a polynomial. Thus, to solve the friend recommendation problem using these protocols, we need to generate a global polynomial (in encrypted form) by combining the polynomials corresponding to the friend lists of A's friends. Let $f_1, \ldots, f_m$ denote the polynomials corresponding to $B_1, \ldots, B_m$, respectively. According to [44], the computation of f (denoting the global polynomial) needs to be done sequentially. That is, B_1 initially sends the encrypted f_1, denoted $E_{pk}(f_1)$ to A (serving as a router), where $E_{pk}(f_1) = \{E_{pk}(f_1[d]), \ldots, E_{pk}(f_1[0])\}$, E denotes the encryption function with public key pk, $f_1[j]$ is a coefficient and d denotes the degree of the polynomial f_1, for $0 \leq j \leq d$. Then A sends $E_{pk}(f_1)$ to B_2, and B_2 returns $E_{pk}(f_1 \cdot f_2)$ to A and so on. After computing $E_{pk}(f)$, more computation is needed to actually identify the element in the union above the threshold. That is, the polynomial f need to be evaluated by each B_i based on his or her private dataset. It is clear that the above solution is complex. Also, we observed that the computation, communication and round complexities of this solution are much higher than our proposed protocols (detailed complexity of the proposed protocols is given in Section 5.3 and Section 6).

Additionally, the evaluation of $E_{pk}(f)$ by B_i may not produce the correct result if A randomizes $E_{pk}(f)$ in order to hide the actual user IDs. On the other hand, if A does not randomize $E_{pk}(f)$, 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 we adopt the aforementioned secure set operation protocols. The protocols proposed in this paper are not only efficient, but also better in terms of protecting the user's privacy and accuracy compared to the solutions based on the existing secure set operation protocols.

3 Threat Model

We briefly summarize the literature work on Secure Multiparty Computation along with the security definition adopted in this paper.

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, we adopt the security definitions in the literature of Secure Multiparty Computation (SMC) first introduced by Yao's

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

Table 1: COMMON NOTATIONS

Millionaire problem for which a provably secure solution was developed [78,79]. This was extended to multiparty computations by Goldreich et al. [31]. It was proved in [31] 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 [4,8,29,43].

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 paper, to develop secure and efficient protocols, we assume that parties are semi-honest. Additional justifications for assuming the semi-honest adversarial behavior are provided in Section 5.4. The following definition captures the above discussion regarding a secure protocol under the semi-honest model [29].

Definition 1. Let a_i be the input of party i, $\prod_i(\pi)$ be i's execution image of the protocol π and b_i be the output computed for party i from π. Then π is said to be secure if $\prod_i(\pi)$ can be simulated from $\langle a_i, b_i\rangle$ such that the 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. To prove a protocol is secure, we generally need to show that the execution image of a protocol does not leak any information regarding the private inputs of the participating parties [29].

4 Preliminaries

In this section, we present some concepts and properties related to universal hash functions, additive homomorphic encryption schemes, and common neighbors method, as a background knowledge, which are used throughout the paper. Some common notations that are extensively used in this paper are highlighted in Table 1.

4.1 Universal Hash Function

Consider a set of integers $L = \{0, 1, \ldots, \ell - 1\}$. The goal is to map the integers from domain L to a smaller domain $V = \{0, 1, \ldots, s - 1\}$ (where $s < \ell$) with minimum number of collisions. This can be achieved by a universal hash function [10]. Given a positive integer $x \in L$, a universal hash function $h_{a,b}$ is defined as follows:

$$h_{a,b}(x) = (a \cdot x + b \bmod p) \bmod s \tag{2}$$

where p is a prime $\geq \ell$, 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 following property: $h(x) - h(y) \bmod s$ is uniformly distributed in $V, \forall x, y \in L$. That is the probability of collision between x and y is $\frac{1}{s}$ (note that there is no perfect hash function without collisions).

4.2 Additive Homomorphic Probabilistic Encryption

In the first proposed protocol, we use an additive homomorphic encryption scheme (denoted as HEnc) that is probabilistic in nature. We now discuss some interesting properties exhibited by the HEnc system. 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[1] 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: $E_{pk}(x_1) \cdot E_{pk}(x_2) = E_{pk}(x_1 + x_2)$,
- 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 [34]. 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, in this paper, we use the Paillier cryptosystem [63] due to its efficiency.

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 by Φ) for A and C is defined as the number of friends/neighbors that A and C have in common [59]. More formally,

$$\Phi(A, C) = |Fr(A) \cap Fr(C)| \tag{3}$$

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

[1] The product of two large prime numbers of similar bit-length.

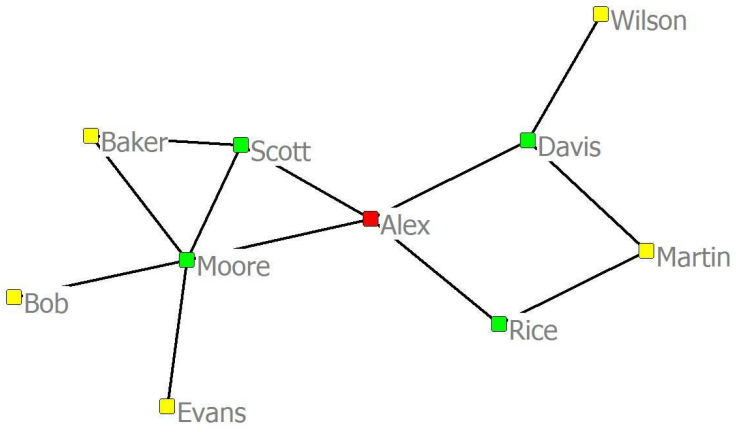


Figure 1: Sample snapshot (two-hop) of a social network for $Alex$

Example 1. Consider the sample two-hop snapshot of the social network for $Alex$ as shown in Figure 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 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 2. Since $\Phi(Alex, Baker) = 2$ and $\Phi(Alex, Martin) = 2$, $Baker$ and $Martin$ are recommended as new friends to $Alex$.

5 The Proposed PPFR Methods

To solve the PPFR problem, we make the following practical assumptions supported in many on-line 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**: Suppose we want to recommend new friends to A based on the common neighbors method discussed in Section 4.3. The 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, the value of t can be decided by either the network administrator or A, and it can be static or changed occasionally. In either case, it is important that the value of t is chosen at random under the assumption that $1 \leq t \leq UNF$, where UNF denotes the upper bound on the number of friends for any given user in the social network. Note that the value of t does not reveal any information about a user's friend list. 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 paper, we simply assume that t is chosen by the network administrator from $[1, UNF]$ and public.
4. **Semi-honest Behavior**: We assume that users as well as the OSN provider are semi-honest (i.e., they do not collude). However, in the case of malicious model, we believe

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

Table 2: Friend lists and the common neighbors scores for $Alex$ based on Figure 1

that the proposed protocols can be combined with the techniques from [25] in order to mitigate the server equivocation problem mentioned in Section 2.2.

In addition to the above assumptions, we assume a social network where 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 [3, 12, 37, 39, 53, 72, 75], have been proposed to protect users' privacy through cryptography. More details about such a network are provided in Section 7.

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 paper does not address these issues when presenting and analyzing the proposed protocols.

5.1 The PPFR$_h$ Protocol

The overall steps involved in the first proposed protocol, denoted by PPFR$_h$, are shown in Algorithm 1. The PPFR$_h$ protocol is based on an additive homomorphic encryption scheme as explained in Section 4. 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, we could use this global space to represent the candidate space without disclosing any particular candidate's ID to A. However, 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. Here we need to have a proper 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, we can map a single digit d ($0 \leq d \leq 9$) to $128 + d$. Then we can append 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 our 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, we simply use A to 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. We 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 [63]. T publishes the public key pk and threshold t to the users in the network. In our protocol, T is only responsible for certain intermediate computations. We emphasize that 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. Each user B_i generates an encrypted matrix that contains his/her friend list information and forwards it to A. Then A aggregates the encrypted friend lists component-wise to get the encrypted frequency for each friend of A's friend. After this, with the help of T who holds the private key pk, A securely retrieves the IDs of the friends of A's friends whose frequency is greater than or equal to threshold t.

Having discussed the basic idea of PPFR$_h$, we now provide a more detailed explanation of each step in PPFR$_h$ which are also outlined in Algorithm 1. 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) 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. The first column of M_i contains user IDs and the second column contains either 0s or 1s. Given the j^{th} row in M_i, if the value of 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. More specifically,

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

where $Fr(B_i)[j]$ denotes 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

Algorithm 1 PPFR$_h$

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

1: A:

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

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

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

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

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[2] on all the encrypted matrices, i.e., M_i''s component-wise and computes a new matrix Z, for $1 \leq j \leq s$, as shown below:

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

[2]Note that the multiplication and exponentiation operations on ciphertexts are followed by $\bmod N^2$ operation so that the resulting ciphertext is still in $\{0, 1, \ldots, N^2 - 1\}$. However, to make the presentation more clear, we simply drop "$\bmod N^2$" from the computations in the rest of this paper.

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). More specifically, T computes $freq_j = D_{pr}(\hat{Z}[j][2])$ and decrypts the first component of j^{th} row of $\hat{Z}$, for $1 \leq j \leq s$, and proceeds 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. Note that the decryption of the first component of each row in $\hat{Z}$ (i.e., $D_{pr}(\hat{Z}[j][1])$) always yields a random value.
- In case of no collision at location $\pi^{-1}(j)$ and $freq_j \geq t$, we can observe that $\beta_j = r_k + \text{ID}_k$, where $k = \pi^{-1}(j)$ and ID_k is one of the recommended friend IDs to A. On the other hand, when $freq_j < t$, β_j is always 0.
- After this, T sends β to A, where $\beta = \langle \beta_1, \ldots, \beta_s \rangle$.

Finally, A applies the inverse permutation on β and removes the randomness for each non-zero entry. Precisely, A selects the list of recommended friend IDs as shown below:

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

We emphasize that the proposed PPFR$_h$ protocol reveals S and $freq_j$ as the final outputs to A and T, respectively, for $1 \leq j \leq s$. Remember that $freq$ denotes the randomly permuted list of common neighbors scores and T cannot link the scores to user identities due to random permutation by the target user A.

Example 2. Refer to Figure 1, let $t = 2$ and $s = 9$. From Figure 1, we have $Fr(Alex) = \langle Scott, Moore, Rice, Davis \rangle$. 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 3. Let M_1, M_2, M_3 and M_4 denote the matrices generated by $Scott$, $Moore$, $Rice$ and $Davis$, respectively, based on the PPFR$_h$ protocol. The intermediate results during various steps involved in PPFR$_h$ are shown in Table 3. At the end of the PPFR$_h$ protocol, $Baker$ and $Martin$ are recommended as new friends to $Alex$.

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, we present 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 key steps involved in PPFR$_{sp}$ are demonstrated in Algorithm 2 and Algorithm 3. The main process of the protocol is presented in Algorithm 2. We assume that each user A in the network generates a public-private key pair (pk_A, pr_A) using the RSA public key system [65]. Also, each friend of an A's friend (say user C_{ij}) uses an AES encryption algorithm [62] to encrypt his/her data, and the secret key is divided into at most $|Fr(C_{ij})|$ different shares such that at least t shares are required to reconstruct the secret AES key [69].

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

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

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

Table 3: Various intermediate results during the execution of the PPFR$_h$ protocol

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 $Fr(A)$. In addition, since A cannot trace back to the candidate, the privacy of each candidate (i.e., friends of A's friends) is preserved.

Next, we explain the overall steps involved in PPFR$_{sp}$ in detail. 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 [42], 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, shown as step 3 in Algorithm 2, C_{ij} receives the message and then appends $k_{C_{ij}}^{B_i}$ (the share of the key corresponding to B_i) and 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 3, 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 (i.e., $k_{C_{ij}}$) 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 [69] 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. We emphasize that two random users are necessary; otherwise, if only one random user were chosen to do the favor, he or she would know that C_{ij} and A share a friend. To increase security, we 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 2. According to the theory in [69], 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, and 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. 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}.
We emphasize that the proposed PPFR$_{sp}$ protocol reveals $|G_{ij}|$ and $\langle \text{ID}(C), \Phi(A, C)\rangle$ as the final output to A, for each user $C \in S$, $1 \leq i \leq m$ and $1 \leq j \leq |Fr(B_i)|$. Remember that $\Phi(A, C)$ denotes the common neighbors score between A and C.

Example 3. Consider the snapshot of the social network for $Alex$ shown in Figure 1, and let $t = 2$. The set of potential candidates for $Alex$ are $\langle Baker, Bob, Evans, Martin, Wilson\rangle$. Following from PPFR$_{sp}$, 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 $Baker$. Also, $Moore$ 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 (different) encrypted paths from $Scott$ and $Moore$, $Baker$ appends his self introductions $k_B^S||\text{AES}_{k_B}(Baker)$ and $k_B^M||\text{AES}_{k_B}(Baker)$ to the respective (random) 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 $\text{AES}_{k_B}(Baker)$. Since $Alex$ has two different key shares i.e., k_B^S and k_B^M corresponding to the group $\text{AES}_{k_B}(Baker)$, he

Algorithm 2 PPFR$_{sp}$

Require: $\forall$ user A, he/she holds the RSA private key pr_A and shares the public key pu_A with 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)

can generate the key k_B (as $t = 2$) and decrypts $\text{AES}_{k_B}(Baker)$ successfully and identifies $Baker$ as the newly recommended friend. Similarly, $Alex$ identifies $Martin$ as a new friend. The rest of the potential candidates are not recommended to $Alex$ since their respective group sizes are less than 2.

Algorithm 3 Collect_and_Solve(t)

$G = \{k_{C_{ij}}^{B_i} || \text{AES}_{k_{C_{ij}}}(C_{ij}), 1 \leq i \leq m \wedge 1 \leq j \leq |Fr(B_i)|\}$
Partition G into G_{ij}'s, where
$G_{ij} = \{k_{C_{ij}}^{B_{i_1}} || \text{AES}_{k_{C_{ij}}}(C_{ij}), \ldots, k_{C_{ij}}^{B_{i_z}} || \text{AES}_{k_{C_{ij}}}(C_{ij})\}$
$S = \emptyset$
for each G_{ij} **do**
  if $|G_{ij}| \geq t$ **then**
    Generate $k_{C_{ij}}$ via polynomial interpolation on $\{k_{C_{ij}}^{B_{i_1}}, \ldots, k_{C_{ij}}^{B_{i_z}}\}$
    Use $k_{C_{ij}}$ to decrypt $\text{AES}_{k_{C_{ij}}}(C_{ij})$ and get C_{ij}
    $S \leftarrow S \cup C_{ij}$
  end if
end for
return S

5.3 Complexity Analysis

5.3.1 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 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 $|Fr(A)|$. On the other hand, 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, the computation time mainly depends on the number of public key encryptions (which depends on the size of each B_i's friend list). This is 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 encryption costs. Therefore, the computation cost of the PPFR$_{sp}$ protocol is bounded by O($|Fr(B_i)|$) encryptions assuming that t and $|Fr(A)|$ are not exceedingly larger than $|Fr(B_i)|$.

5.3.2 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 whose size 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$_{sp}$, the 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 we have $\Sigma_{i=1}^{m}|Fr(B_i)|$ number of (B_i, C_{ij}) pairs, the total communication complexity of PPFR$_{sp}$ is bounded by $O(K \cdot w)$, where $w = \Sigma_{i=1}^{m}|Fr(B_i)|$. 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., follow the prescribed steps of the protocol.

5.4 Security Analysis

5.4.1 The Semi-Honest Security Model

We have 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 Multiparty Computation (SMC) are under this model. By using standard zero-knowledge proofs [32] for threshold Paillier homomorphic encryption scheme [11, 16], 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. We restrict our discussion to the semi-honest model for the rest of this section.

5.4.2 Security Analysis under the Semi-Honest Model

To prove the security of our proposed protocols, we adopt the standard proof methodology in the literature of SMC. According to Definition 1, we 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, we can claim that the protocol is secure [29]. We emphasize that the security of participating parties in the proposed protocols 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, we 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, we need to show 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 1, Π_A contains $\{h_{a,b}, pk, M'_1, \ldots, M'_m, S\}$. 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 [63], $M'_i[j][k]$ is computationally indistinguishable from $R_i[j][k]$ [33]. 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, we can prove that the PPFR$_h$ protocol is secure from user A's perspective. That is, A's friend list is not disclosed to T. We emphasize that, in PPFR$_h$, T knows the frequency information of each permuted and randomized user ID (i.e., $freq_j$ in Algorithm 1). However, $freq_j$ is considered to be the output of T in PPFR$_h$. As a result, according to Definition 1, it is clear that the execution image of T can be easily simulatable as it depends only on its output $freq_j$.

We stress that it is also possible to completely eliminate revealing $freq_j$ to T by first adopting a fast secure binary conversion protocol over encrypted data [67], and then securely comparing the encrypted frequency of each potential candidate with the threshold t without ever disclosing the frequency information to T. In this way, the PPFR$_h$ offers the best security. The price to pay here is the increased computation cost. Nonetheless, since T does not know the hash function, the additional information that T can deduce from $freq_j$ is extremely small. Hence, the current implementation of PPFR$_h$, which assumes that $freq_j$ is the output of T, is likely sufficient.

In PPFR$_{sp}$, unlike PPFR$_h$ where the scores are revealed to T, the common neighbors scores (i.e., $|G_{ij}|$ in Algorithm 3) are directly revealed to A. Also, A can 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 [69]. Since all the above information is considered to be the output of A, the execution image of A can be easily simulatable from its input and output according to Definition 1. Furthermore, other than the intermediate results (which are in encrypted format), no additional information is revealed to B_i. In general, PPFR$_h$ is more secure than PPFR$_{sp}$, but it is less efficient.

5.4.3 Inference Problem and the Ideal Security Model

In the proposed protocols, there exists an inference problem, but we also like to point out that the inference problem exists even for the most secure common-neighbor based PPFR protocol. Our 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 our problem domain). 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 [29]. 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 [29].

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. 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/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 in the protocol to prevent the content of his or her friend list from being inferred.

6 Empirical Analysis

In this section, we thoroughly analyze the effectiveness and computation cost of both the proposed protocols using a Facebook dataset. Since the PPFR$_{sp}$ protocol always recommends friend(s) accurately, we present the effectiveness of PPFR$_h$ by using the PPFR$_{sp}$ protocol as a baseline. In addition, the computation costs for both protocols are analyzed under different parameter settings.

6.1 Dataset and Platform Description

For our experimental setting, we collected friend lists of 11,500 Facebook users from Facebook.com directly 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 we can extract the friend list using a proper regular expression. Remember that Facebook lets users to set their privacy preferences. Thus, we can only crawl those users who set their friend lists information as public.

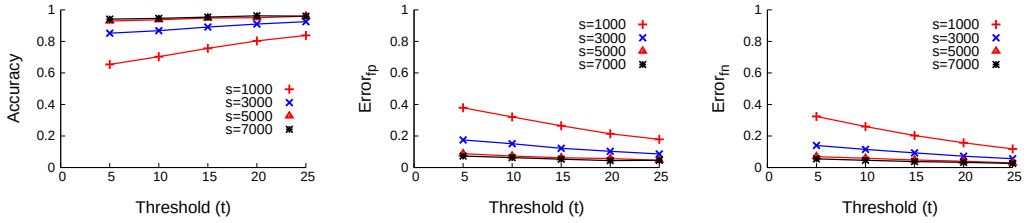
Due to privacy settings, some of the extracted friend lists are empty (for 2,068 users). In addition, the combined friend lists of users' friends are empty for 112 users. Therefore, after removing the friend lists of these users, we finally had friend lists for 9,330 users. Among the 9,330 users, the 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 LTS, with 12GB memory.

6.2 Effectiveness of PPFR$_h$

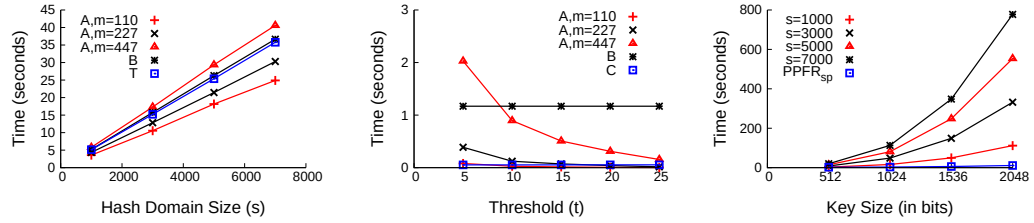
We analyze the effectiveness of PPFR$_h$ by using PPFR$_{sp}$ as the baseline. Out of the 9,330 Facebook users, we selected 1,000 users randomly as target users and conducted various experiments based on different parameters. We denote the target users (who wish to find new friends) as set $\mathcal{D}$. Hereafter, the effectiveness results presented are average values over the 1,000 random users in $\mathcal{D}$. (We observed that our 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, we define the accuracy of PPFR$_h$ (with respect to PPFR$_{sp}$) as the common ratio between the two sets as follows:

$$\text{Accuracy} = \frac{|S_h \cap S_{sp}|}{|S_{sp}|} \tag{4}$$



(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 2: Empirical results for PPFR$_h$ and PPFR$_{sp}$

Note that S_{sp} always contains the accurate friend recommendations. We compute accuracy for each user in $\mathcal{D}$ according to different hash/vector sizes (s) and threshold values (t). As shown in Figure 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 we increase the value of s from 1,000 to 7,000. The reason behind is that as we increase the value of s, the number of collisions are reduced; therefore, improving the accuracy. We also observe that, as shown in Figure 2(a), for a fixed value of s, the accuracy improves with an increase in the value of t. For example, when $s = 5,000$, the accuracy of PPFR$_h$ changes from 92.1% to 96% as we increase t from 5 to 25. A similar trend can be observed for other values of s and t. The 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 the intersection size, we also analyzed 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 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%. Also, for a fixed t, we observe that the false positive error rate decreases when s increases (resulting in less hash collisions). For example, when $t = 25$, false 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 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 $s = 7,000$ 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 we illustrate the computation costs incurred for different parties involved in the PPFR$_h$ and PPFR$_{sp}$ protocols separately. We also compare the running time of both protocols for different values of s and key sizes.

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). On the other hand, the computation cost of each friend of A (i.e., B) mainly depends on s (the deciding factor for the number of encryptions to be performed by each B). For a friend B of A, we observed 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). For the PPFR$_h$ protocol, the computation costs of A and B are independent from t.

Based on the above discussions, we now present the computation costs of each party in the PPFR$_h$ protocol. Since the friend list sizes vary for different users, we have 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. We emphasize that similar results can be observed for other users. Note that in our Facebook dataset, 447 is the maximum friend list size. For key size 1,024 bits, we ran the PPFR$_h$ protocol for each of these three users by varying the values of s.

As shown in Figure 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 2(d), the computation cost of B and T are the same under the assumption that T has to decrypt the whole encrypted matrix $\hat{Z}$ (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 (independent of m), respectively. Our 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).

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 our experiments, we used the Shamir's secret sharing scheme [69] to generate the secret partial keys or shares.

We 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 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 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). In addition, we consider the worst case for computing C's time (assuming 447 friends). 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 took 53 milliseconds for C to create all 447 secret shares and also to encrypt his/her ID using the AES private key.

6.5 The Computation Cost of PPFR$_h$ vs. PPFR$_{sp}$

We also compared the total running time of both protocols for the worst case (i.e., A with 447 friends) by varying the values of key size and s, and the results are shown in Figure 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 running 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. Our results also 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 [19, 30] 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 users' friend lists are private.

7 Implementation Details

In this section, we point 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, we consider the privacy-enhanced web pages where a user can encrypt his/her profile information before sending it to the OSN [3]. That is, the data stored on the OSN server is encrypted using the 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, our proposed protocols make friend recommendation possible under the above mentioned architecture. In order to use the proposed protocols in a full fledged manner, we first need to address an issue that arises due to the inherent information flow in OSNs which we discuss below.

In the proposed protocols, we assume that users can directly exchange messages 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 intended to receive the message may not be online. In addition, in order to minimize the information disclosure to T, we need to make sure that the messages exchanged between users are in encrypted form and only the intended receiver is able to decrypt it.

The above issue can be solved by creating a secure session between the users. Briefly, we 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, our 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. We emphasize that the proposed protocols, after taking the above implementation details into account, protect the friend lists of a user from other users as well as from T.

8 Conclusions

The emerging growth of social networks has resulted in vast amount of social data which need to be mined for various kinds of recommendations, including friend recommendation. However, since the social data contain personal and sensitive information about individual users, the existing friend recommendation techniques do not work when the users' friend lists remain private. As a result, this paper proposes two privacy-preserving friend recommendation algorithms under the assumption that users' friend lists should be kept private. Both of the proposed protocols recommend new friends using the common neighbors proximity measure in a privacy-preserving manner.

The first proposed protocol, denoted by PPFR$_h$, is based on an additive homomorphic encryption scheme, and its accuracy is essentially based on the parameters of universal hash function $h_{a,b}$. The second proposed protocol, denoted by PPFR$_{sp}$, utilizes the concept of protecting the source privacy through anonymous message routing and also recommends friends accurately. We observe that PPFR$_{sp}$ is more efficient than PPFR$_h$, but it is less secure. The proposed PPFR protocols act as a trade-off among security, efficiency and accuracy.

In the PPFR$_{sp}$ protocol, if both A and his/her friends have long friend lists, then the pro-

tocol will incur a lot of communication among the participating 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. It is also possible that a new friend can be recommended in other ways; for example, by checking the similarities between the users' profile contents or activities in the network. Therefore, we will explore alternative ways to develop a hybrid privacy-preserving friend recommendation method by combining different scoring functions in our future work. Furthermore, we will investigate the side effects of adding new friends who were recommended due to false positives in the first proposed protocol.

Acknowledgments

We would like to thank the anonymous reviewers for their invaluable feedback and suggestions. This work has been partially supported by the National Science Foundation under grant CNS-1011984.

References

[1] 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.

[2] 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. IEEE Computer Society, 2010.

[3] 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.

[4] A. Ben-David, N. Nisan, and B. Pinkas. Fairplaymp - a system for secure multi-party computation. In *Proceedings of the ACM Computer and Communications Security Conference (CCS)*. ACM, 2008.

[5] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.

[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 (TIST)*, 2:22:1–22:37, May 2011.

[7] D. Boyd and N. B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, October 2007.

[8] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000.

[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. ACM, 2009.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*, chapter Data Structures, pages 229–277. The MIT Press, Cambridge, MA, USA, 3rd edition, 2009.

[11] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology – EUROCRYPT*, pages 280–299. Springer, 2001.

[12] 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. IEEE Computer Society, 2012.

[13] L. A. Cutillo, R. Molva, and M. Onen. Analysis of privacy in online social networks from the graph theory perspective. In *Proceedings of the Global Telecommunications Conference (GLOBECOM)*, pages 1 –5. IEEE, 2011.

[14] L. A. Cutillo, R. Molva, and T. Strufe. Safebook: Feasibility of transitive cooperation for privacy on a decentralized social network. In *the International Symposium on a World of Wireless, Mobile and Multimedia Networks Workshops (WOWMOM)*, pages 1–6. IEEE, 2009.

[15] B.-R. Dai, C.-Y. Lee, and C.-H. 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)*, pages 709 –714. IEEE Computer Society, 2011.

[16] I. Damgård, M. Jurik, and J. B. Nielsen. A generalization of paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9(6):371–385, December 2010.

[17] 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)*, pages 39–44. ACM, 2010.

[18] W. Dong, V. Dave, L. Qiu, and Y. Zhang. Secure friend discovery in mobile social networks. In *Proceedings of INFOCOM*, pages 1647–1655. IEEE, 2011.

[19] 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)*, pages 13–22. ACM, 2001.

[20] C. Dwork. Differential privacy. In *Proceedings of ICALP*, pages 1–12. Lecture Notes in Computer Science, Springer, 2006.

[21] C. Dwork. Differential privacy: a survey of results. In *Proceedings of the 5th international conference on Theory and applications of models of computation (TAMC)*, pages 1–19. Springer-Verlag, 2008.

[22] 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)*, page 339. Association for Information Systems, 2007.

[23] D. M. Ehrlich. Social network survey paper. *International Journal of Learning and Intellectual capital*, 3:167–177, January 2006.

[24] Facebook. Official blog: `http://blog.facebook.com/blog.php?post=15610312130`, 2008.

[25] 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*, pages 31–31. USENIX Association, 2012.

[26] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Eurocrypt*, pages 1–19. Springer, 2004.

[27] K. Frikken. Privacy-preserving set union. In *Applied Cryptography and Network Security*, volume 4521, pages 237–252. Lecture Notes in Computer Science, Springer, 2007.

[28] H. Gao, J. Hu, T. Huang, J. Wang, and Y. Chen. Security issues in online social networks. *IEEE Internet Computing*, 15(4):56 –63, July/August 2011.

[29] O. Goldreich. *The Foundations of Cryptography*, volume 2, chapter Encryption Schemes, pages 373–470. Cambridge University Press, Cambridge, England, 2004.

[30] O. Goldreich. *The Foundations of Cryptography*, volume 2, chapter General Cryptographic Protocols, pages 599–746. Cambridge, University Press, Cambridge, England, 2004.

[31] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. In *19th Symposium on the Theory of Computing*, pages 218–229. ACM, 1987.

[32] 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, July 1991.

[33] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[34] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, 18:186–208, February 1989.

[35] L. Gou, F. You, J. Guo, L. Wu, and X. L. Zhang. Sfviz: interest-based friends exploration and recommendation in social networks. In *Proceedings of Visual Information Communication - International Symposium (VINCI)*, pages 1–10. ACM, 2011.

[36] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the workshop on Privacy in the electronic society (WPES)*, pages 71–80. ACM, 2005.

[37] S. Guha, K. Tang, and P. Francis. Noyb: privacy in online social networks. In *Proceedings of the first workshop on Online social networks (WOSN)*, pages 49–54. ACM, 2008.

[38] 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.

[39] 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)*, pages 411–415. ACM, 2011.

[40] 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.

[41] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth conference on Recommender systems (RecSys)*, pages 135–142. ACM, 2010.

[42] W. Jiang, L. Si, and J. Li. Protecting source privacy in federated search. In *Proceedings of the 30th annual international conference on Research and development in information retrieval (SIGIR)*, pages 761–762. ACM, 2007.

[43] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman & Hall, CRC Press, 2007.

[44] L. Kissner and D. Song. Privacy-preserving set operations. In *Advances in Cryptology - CRYPTO*, pages 241–257. Springer, 2005.

[45] 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)*, pages 289–298. ACM, 2008.

[46] B. Krishnamurthy and C. E. Wills. Characterizing privacy in online social networks. In *Proceedings of the first workshop on Online social networks (WOSN)*, pages 37–42. ACM, 2008.

[47] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th international conference on Knowledge discovery and data mining (SIGKDD)*, pages 611–617. ACM, 2006.

[48] 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.

[49] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the 12th international conference on Information and knowledge management (CIKM)*, pages 556–559. ACM, 2003.

[50] 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, May 2007.

[51] 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*, pages 61–80. Springer-Verlag, 2011.

[52] 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)*, pages 121–128. IEEE Computer Society, 2006.

[53] 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)*, pages 1–8. ACM, 2008.

[54] H. Ma, T. C. Zhou, M. R. Lyu, and I. King. Improving recommender systems by incorporating social contextual information. *ACM Transactions on Information Systems (TOIS)*, 29:1–23, April 2011.

[55] A. Machanavajjhala, A. Korolova, and A. D. Sarma. Personalized social recommendations: accurate or private. *Proc. VLDB Endowment*, 4:440–450, April 2011.

[56] J. P. Mello. Facebook scrambles to fix security hole exposing private pictures, December 2011. `http://www.pcworld.com/article/245582/facebook_scrambles_to_fix_security_hole_exposing_private_pictures.html`.

[57] 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. ACM, 2007.

[58] 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. IEEE, 2011.

[59] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review Letters E*, 64:025102, July 2001.

[60] 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.

[61] 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)*, pages 337–348. ACM, 2012.

[62] NIST. Advanced encryption standard. Technical Report NIST Special Publication FIPS-197, National Institute of Standards and Technology, 2001.

[63] 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.

[64] D. Project. `http://diasporaproject.org/`.

[65] 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, February 1978.

[66] B. K. Samanthula and W. Jiang. Structural and message based private friend recommendation. In *Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 684–690. IEEE Computer Society, 2012.

[67] B. K. Samanthula and W. Jiang. An efficient and probabilistic secure bit-decomposition and its application to range queries over encrypted data. In *Proceedings of the 8th ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 541–546. ACM, 2013.

[68] A. Shakimov, H. Lim, R. Caceres, L. Cox, K. Li, D. Liu, and A. Varshavsky. Vis-à-vis: Privacy-preserving online social networking via virtual individual servers. In *Third International Confer-*

ence on Communication Systems and Networks (COMSNETS), pages 1–10. IEEE, 2011.

[69] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[70] 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. IEEE, 2010.

[71] E. Steel and J. E. Vascellaro. Facebook, myspace confront privacy loophole. The Wall Street Journal, May 2010. `http://online.wsj.com/news/articles/SB10001424052748704513104575256701215465596`.

[72] 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)*, pages 2516–2524. IEEE, 2010.

[73] 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)*, pages 1–6. ACM, 2008.

[74] 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)*, pages 61–68. ACM, 2011.

[75] 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)*, pages 169–180. ACM, 2009.

[76] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove. An analysis of social network-based sybil defenses. In *Proceedings of the ACM SIGCOMM conference*, pages 363–374. ACM, 2010.

[77] X. Xie. Potential friend recommendation in online social network. In *IEEE/ACM International Conference on Cyber, Physical and Social Computing and International Conference on Green Computing and Communications*, pages 831 –835. IEEE Computer Society, 2010.

[78] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE Computer Society, 1982.

[79] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society, 1986.

[80] 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.