Privacy Preserving Friend Discovery in Mobile Social Networks

by Hongjuan Li

B.S. in Computer Science and Technology, July 2008, Dalian JiaotongUniversity
M.S. in Computer Applied Technology, July 2011, Dalian University of Technology

A Dissertation submitted to

the Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Doctor of Philosophy

January 31^{th}, 2016

Dissertation directed by

Xiuzhen Cheng
Professor of Computer Science

The School of Engineering and Applied Science of The George Washington University certifies that Hongjuan Li has passed the Final Examination for the degree of Doctor of Philosophy as of December 8th, 2015. This is the final and approved form of the dissertation.

Privacy Preserving Friend Discovery in Mobile Social Networks

Hongjuan Li

Dissertation Research Committee:

Xiuzhen Cheng, Professor of Computer Science, Dissertation Director

Hyeong-Ah Choi, Professor of Computer Science, Committee Member

Nan Zhang, Associate Professor of Computer Science, Committee Member

Hang Liu, Associate Professor of Electrical Engineering & Computer Science, The Catholic University of America, Committee Member

Abstract

Privacy Preserving Friend Discovery in Mobile Social Networks

Mobile social networking has been increasingly popular with the explosive growth of mobile devices. By allowing mobile users to interact with potential friends around the real world, it enables new social interactions as a complement to web-based online social networks. Motived by this feature, many exciting applications have been developed, yet the challenge of privacy protection is also aroused. This dissertation studies the problem of privacy preserving in mobile social networks. We propose different mechanisms for various privacy requirements.

The first algorithm we proposed is a secure friend discovery mechanism based on encounter history in mobile social networks, which mainly focuses on the *location privacy*. By exploring the fact that sharing encounters indicate common activities and interests, our scheme can help people make friends with like-minded strangers nearby. We provide peer-to-peer confidential communications with the location privacy and encounter privacy being strictly preserved. Unlike most existing works that either rely on a trusted centralized server or existing social relationships, our algorithm is designed in an ad-hoc model with no such limitation. As a result, our design is more suitable and more general for mobile social scenarios.

We also develop an efficient customized privacy preserving mechanism, which not only protects the privacy of users' profile, but also establishes a verifiable secure communication channel between matching users. Besides, the initiator has the freedom to set a customized request profile by choosing the interested attributes and giving each attribute a specific value. Moreover, the request profile's privacy protection level is customized by the initiator according to his/her own privacy requirements. We also consider the collusion attacks among unmatched users. To the best of our knowledge, this is the first work to address

such security threat. Our protocol guarantees only exactly matching users are able to communicate with the initiator securely, while as little as possible information can be obtained by other participants. To increase the matching efficiency, our design adopts the Bloom filter to efficiently exclude most unmatched users. As a result, our design effectively protects the profile privacy and efficiently decreases the computation overhead.

Our third work for this dissertation explores fine-grained profile matching by associating a user-specific numerical value with each attribute to indicate the level of interest. And the profile similarity is computed with a secure dot-product. While existing studies are mainly focused on leveraging rich cryptography algorithms to prevent privacy leakage, we consider a novel cooperative framework by mixing some random noise with the private data to preserve privacy. By carefully tuning the amount of information owned by each party, we guarantee that the privacy is effectively preserved while the matching result of two profiles can be cooperatively obtained. After giving an introduction of the basic mechanism, we detail two enhanced mechanisms by taking collusion attack and verifiability into consideration. With no expensive encryption algorithms involved, our methods are more practical for real-world applications.

Table of Contents

List of Figures

List of Tables

Chapter1

Introduction

1.1 Private Profile Matching in MSNs

Mobile social networks (MSNs) [31, 40] have been introduced by combining the concepts from two disciplines, i.e., social networks and mobile communications. The notion of social networks has been used in the context of information and communication technologies to provide data exchange, sharing and delivery services by exploring the social relations [58]. Due to the ubiquitous availability of mobile devices (such as smart phones), mobility has become an integral part of our human society. However, mobility results in random behavior of users, making it difficult for data exchange, sharing, and delivery services [10].

A mobile social network can fully take advantage of human interaction and physical mobility to achieve efficient and effective data delivery services. A mobile social network can be established on any existing centralized or distributed mobile networks [39, 88].

- Centralized MSNs

  Centralized MSNs are mainly an extension of their web-based predecessors. All the information concerning the members of a social network are preserved in remote servers belonging to a service provider. By using specific mobile applications or simply by using a mobile browser, users are able to connect to a centralized server via cellular network, WiFi, or other similar technologies. In other words, users will access the remote server in order to communicate with each other for sharing and exchanging information.

- Distributed MSNs

  The key feature of the distributed MSNs is the absence of centralized servers [74].

Mobile users are able to communicate and access social information by connecting to other users. Such connectivity is best defined by opportunistic contacts [32,45,58] where users exchange and share information whenever they come in contact using wireless technologies such as Bluetooth or Wi-Fi. The mobile terminals nowadays are equipped with multiple wireless interfaces, providing them with the capability of connecting in an ad-hoc manner [37], with no need for any kind of infrastructure.

Our research work mainly focuses on the distributed mobile social networks. With this type of MSN, the data are generated by mobile users, and these users interact with each other on the basis of their common interests thus establishing social relations with potential friends [74].

By allowing mobile users to interact with potential friends around the real world, mobile social networks [4, 35, 56] enable new social interactions as a complement to the web-based online social networks. Physically proximate mobile users can directly communicate with each other through the Bluetooth/WiFi interfaces embedded in their mobile devices. Compared to Internet-based social networks, such mobile social networks offer a wider range of applications because of the localization capability of mobile phones and the unnecessary access to a computer or the Internet. Due to the explosive growth of mobile-connected devices such as smartphones, tablets, laptops, and eReaders, mobile social networking [12, 76, 77] has become a promising technology [78].

While MSNs give a great promise for many exciting new applications, they also raise serious privacy and security concerns [43, 64, 66, 68]. Most applications facilitate friend discovery based on profile matching, however, the content of profile includes sensitive information, such as personal background, hobbies, careers, etc. Such privacy concerns hinder the wide adoption of many exciting new applications. On one hand, people prefer over strangers to socialize with others who share common interests and background. Such social reality makes *profile matching* the first step towards effective MSN, which refers to

two users comparing their personal profiles before real interaction. On the other hand, people do not want to disclose their personal profiles to arbitrary persons in physical proximity. And privacy protection is more urgent in mobile social networks as an attacker may be located nearby and able to launch more targeted attacks.

Such situation necessitates *private profile matching*, where two users compare their profiles without revealing the profile privacy to each other. Typical computation of private matching includes two mainstreams: the secure calculation of set intersection and private dot-product. For the first mainstream, most existing work tends to use private set intersection (PSI) [42, 83] or private set intersection cardinality (PSI-CA) [9, 17] for coarse-grained private matching. There are other variations based on PSI/PSI-CA that mainly explore how to improve their efficiency. To differentiate users with the same attributes, the second mainstream of secure computation of dot-product [11, 36, 86] is used for fine-grained private profile matching. In such mechanisms, the similarity between two users' profiles are transferred to dot-product, and homomorphic cryptography [21] is often adopted in the secure computation procedure.

Private matching can be viewed as a special case of secure two-party computation, which can be generalized to secure multi-party computation. Hence, all of these computations can be solved by using the classical secure multi-party computation (SMC) techniques [22]. Sometimes, secret sharing [16, 29, 62] is adopted in multi-party computation for further security enforcement [46]. However, secure multi-party computation suffers from high computation/communication cost, which usually renders these methods impractical for real-world problems. The existing literature on secure multi-party computation thus focuses on devising more efficient solutions for specific functions.

Another common drawback of most of these mechanisms is their lack of result verification: in the final step, one party learns the matching result and then transfers to the other party, resulting in an unverifiable friend discovery.

In addition, the profile matching procedure is performed by all the users in these applications. Only by conducting the matching can the users learn how similar they are. In other words, matched users and unmatched users are all involved in the expensive matching procedure. It is clearly a waste of resources for unmatched users, let alone a more severe problem exists: privacy leakage. Unmatched users also learn the matching results (*e.g.* profile intersection with the initiator), while they are not supposed to.

Recently, a mechanism [85] is proposed to address the privacy preserving verifiable profile matching problem in decentralized MSNs. This work takes advantage of the common attributes between the matched users to generate a key for secure channel construction. However, since the key is based on some common attributes, it violates the randomness requirements for security. To prevent a key from being guessed, keys need to be generated truly randomly and contain sufficient entropy [13].

1.2 General Ideas

Motivated by the limitations in the existing work, we introduce three novel privacy preserving friend discovery mechanisms in this dissertation.

1.2.1 Encounter Based Profile Matching

We believe that users who share common encounters are prone to make friends with each other, because an encounter implies that they may have some common interests or be engaged in similar occupation. For example, people encounter in an NBA final could be big fans of basketball with a high probability; when people encounter in the international conference INFOCOM, probably they are computer scientists. By taking advantage of this fact, we let the mobile devices record the encounter information when two users are within physical proximity. And at a later time, their encounter history should be matched automatically when they encounter again. If the number of encounters is larger than a threshold, or the encounter history satisfies some requirements, the users are recommended

to be friends.

Our encounter history model is valuable in view of the popular "missed connection" services in newspaper and websites like Craigslist. Such a service is similar to Craigslist in that we also provide the service for missed connection. The difference lies in that we let users establish keys when they are in connection, and we take the events in which people encounter as their potential background information. Sharing proximity frequently over time indicates common activities and interests, an important factor for friendship maintenance.

Obviously, the encounter history in a mobile devices includes privacy information since it consists of all the events the device owner has attended, and all the locations the device owner has visited. Such privacy concerns hinder the wide adoption of many exciting new applications. Motivated by this observation, we conduct a privacy-preserving encounter history match. To be more specific, when two users encounter, they establish a shared key, and this specific encounter information is encrypted by the key. Besides, the key serves as a proof in later verification phase that the two users do share an encounter. In this way, each user keeps a record of each encounter, and each single encounter can only be decrypted by the users who did have the same encounter. Except for the common encounters, users know nothing else about each other.

Two users verify to each other they did have an encounter based on their shared key. This shared key is also used to encrypt their follow-up communications. One important thing to note is that each key is limited to two users, which is the shared symmetric key established in a peer-to-peer fashion. This ensures that even there are more than two people in the same particular place and time, the communication between two users is kept confidential from other encounter participants.

1.2.2 Efficient Customized Profile Matching

We propose another novel privacy preserving friend discovery mechanism in our second work. In this protocol, we allow the initiator to construct a customized request profile consisting of a number of attributes. Each attribute has a header indicating its category name and a value field. The attribute headers and the corresponding values are specified by the user. In this way, the user can make customized request based on his/her specific preference. After such a request is sent, a privacy-preserving profile matching is conducted. To reduce unnecessary computation and communication overhead for unmatched users, we employ a filter to exclude most unmatched users. So that only potential matching users will further conduct profile matching instead of all of the received users. For profile matching, our mechanism uses the Attribute Based Encryption (ABE) [5, 23, 59, 69]. Only the users having all the required attributes can decrypt the message. We assume the users who share the requested attributes tend to make friends with each other and construct trust relationships with each other. ABE helps such users with trust establishment while guaranteeing no participants' profile and the submitted preference-profile are exposed to arbitrary users.

Our mechanism is also verifiable, which makes sure both initiator and unmatched user cannot cheat each other by pretending to be matched.

While most existing work only emphasizes privacy preserving but ignores collusion attacks, we further enhance our scheme by taking collusion resistance into consideration.

1.2.3 Fine-grained Profile Matching

The previous two schemes can enable only coarse-grained private matching and cannot further differentiate users with the same attribute(s) but different levels. For example, Alice, Bob and Charlie all have movie as one of their interests. However, the level of this interest is different for them. Alice and Bob go to cinema twice a week, while Charlie may do so

once a month. Apparently, Alice and Bob are more similar in this hobby. However, Bob and Charlie appear the same to Alice. A natural step to solve this problem is fine-grained private matching, where a value is associated with each attribute to indicate the level of this interest. And the profile is transformed into a vector. Then the similarity between two profiles is measured, which is usually converted into a secure dot-product protocol.

We propose a highly efficient mechanism for fine-grained privacy-preserving profile matching [7, 86]. Basically, we do not involve those complex cryptographic algorithms of existing work, instead, a certain amount of noise is added to the private data. And by leveraging the concept of information hiding, the information of each party is strictly limited. By this way the profile matching can be conducted in a carefully designed cooperative manner, so that the privacy of each party can be preserved while the matching result can be obtained.

No server or trusted third party is involved in our model because centralized servers cannot always be relied upon to preserve data confidentiality. We propose a novel cooperative framework in a completely distributed way. Our essential idea is to conduct private profile matching between two users with the help of other users. In the meantime, each party has limited knowledge about the information used for matching while the private profile is only known by the corresponding owner. Note that, providing incentives for the users to cooperate is an important topic, but the discussion of this topic is beyond the scope of this paper. There exist mechanisms [3, 15, 84] addressing the cooperation incentives. A simple motivation could be that a user must be willing to contribute for computation to be able to use this application for friend making.

The standard approach for predicting friendship is based on the closeness or similarity between nodes in a social network. Many quantity measures have been proposed, including the number of common neighbors, the number of common attributes, cosine similarity, and path-ensemble-based measures. They can all be cast as a dot-product operation on two vectors. Therefore, we mainly focus on the computation of secure dot-product without

loss of generality [11]. Since secure dot-product is a fundamental primitive in privacy-preserving data mining and secure multi-party computation, our technique can have many applications beyond mobile social networks.

1.3 Outline

The rest of the proposal is organized as follows. Chapter 2 provides an overview on the related work about privacy concerns in mobile social networks. In Chapter 3, we describe, discuss, and evaluate our first proposed scheme secure friend discovery based on encounter history. In Chapter 4, a detailed description of the second algorithm – efficient customized privacy preserving mechanism is given, the security analysis and performance evaluation are also reported. In Chapter 5, we introduce our third work which is fine-grained privacy preserving profile matching. And we conclude the dissertation in Chapter 6. Plans for future research are presented in Chapter 7.

Chapter2

Related Work

Mobile social networks offer a great promise for enabling many exciting new applications with large potential user bases. They also arouse serious security and privacy concerns. while the mobile devices are location-aware, they can acquire their location via pre-installed cellular/WiFi/GPS positioning software, which imposes the privacy threat: *location privacy*. Besides, the content of a profile includes sensitive information, such as personal background, hobbies, careers, etc, resulting in the *profile privacy* . People are often reluctant to reveal their location/presence information and profile privacy to an arbitrary person in their vicinity. In this chapter, we will review work related to privacy-preserving profile matching in mobile social networks, which includes both *location privacy* and *profile privacy*. We also review the mechanisms for key establishment involved in our work.

2.1 Location Privacy

The field of location privacy has been a very active area of research in recent years. Multiple privacy-preserving mechanisms have been proposed according to different design principles. The most popular protection mechanisms are to decrease the accuracy or precision of the location/time based on anonymization and obfuscation methods, especially to meet the k-anonymity requirement [20,24,38,80]. To obtain a space- or time-obfuscated version of the users' actual location, an anonymizer is adopted, which is a trusted server [19, 24, 30, 38]. Users send their queriers to the anonymizer through a secure connection. Then the anonymizer removes the identities of the users and further constructs a cloaking region such that there are at least k users in this region. In this process, dummy queries that are indistinguishable from the real queries may be added [8].

Unfortunately, the assumption that there is an available trustworthy central server may not be held in practice. According to a recent study [43], all the 13 mobile online social networks under consideration, including Facebook, Friendster, Hi5, LinkedIn, Myspace, and Twitter, leak private information to tracking sites. As a result, people may not be willing to disclose their location information even to a trusted server. Given this situation, Manweiler et al. [52] provide a mechanism with similar k-anonymity guarantees but without requiring the third-party service be trustworthy.

Nevertheless, this server based solution is not suitable for mobile social networks for the following reasons. First, users in a mobile social network may not have direct access to a computer or the Internet. In other words, a central infrastructure is not always present. Second, a server poses an increasing potential risk of becoming an attack bottleneck, a single point of failure, and the target of denial-of-service attacks. Our approach is not subject to these limitations, since we achieve location privacy protection without requiring a third-party server.

2.2 Profile Privacy

To enable friend making while preserving the profile privacy, private profile matching is necessary. Most previous private matching work is based on Private Set Intersection (PSI) [42, 83] or Private Set Intersection Cardinality (PSI-CA) [9, 17], whereby two mistrusting parties with each having a private set jointly compute the intersection or the intersection cardinality of the two sets. Different variants of PSI and PSI-CA have been developed, and they provide the cryptographic foundation for many private matching schemes [46, 48, 50, 57, 75]. These schemes enable coarse-grained private matching and match two users based on their attribute sets' intersection. Such mechanisms are usually tackled with Secure Multi-party Computation (SMC). The general SMC techniques [82] heavily rely on cryptography, and are well-known for their inefficiency. Although various solutions have been proposed to improve efficiency, when applied to the mobile devices,

they are impractical for incurring either high computational or communication overhead.

In addition, these protocols make each pair of users conduct the private matching, which requires multiple rounds of interactions. Even unmatched users are involved in the expensive computation to perform the private profile matching. All users that receive the initiator's query must try to decrypt the message, although most of them hold the wrong keys. This wastes the computation resource and increases the search delay.

Another problem inherent to the PSI based schemes is that the users can learn the profile intersection with the initiator. If several users put their intersection attributes together they may figure out the private profile of the initiator. This would facilitate collusion attacks, which are mostly ignored in the above protocols. Although the PSI-CA mechanisms do not face this problem since they only return a general value indicating the intersection cardinality or profile distance, they cannot support a precise profile matching. In other words, they do not provide one by one matching with the specific attributes proposed by the initiator. For example, they cannot find the user whose profession is "computer engineer" and hometown is "Virginia".

Furthermore, these protocols are unverifiable. The result is known only by one party (say, Alice), who then sends it back to the other party. This relies on the assumption that Alice is trust-worthy, which is not always true in practice. Thus, it is desirable to design a protocol whereby the result is verifiable by both parties.

To enable fine-grained profile matching, Zhang *et al.* [87] propose a protocol that enables finer differentiation among the users having different levels of interest in the same attribute. The profile distance is computed with a secure dot-product. Another similar work using secure dot-product is introduced by Dong *et al.* [11]. The authors presented several techniques and protocols that do not use the intersection (cardinality) of attribute sets for profile matching. Instead, these approaches compute social proximity between two users to discover potential friends, which measures the closeness or similarity between two nodes

in an online social network. Note that they rely on the users' existing relationships in an online social network.

Another work [85] that does not fall into the PSI or PSI-CA category utilizes the common secret shared by the initiator and a matched user. However, this approach uses the request profile as a key; thus only a matched user who shares the secret can decrypt the message. From the perspective of cryptography, the generated key in this work is not truly random and violates the randomness requirements of a secure key.

2.3 Key Establishment

An encounter occurs when two users appear in the physical proximity of each other. In our consideration, this encounter information is also private, which should not be inferred by an unauthorized user who was not present. Lin *et al.* [48] propose a protocol for efficient private proximity testing based on location tags. The goal of private proximity testing is to test whether two mobile users are in close proximity without revealing any additional information about their locations. Since a location tag, which is first introduced by Qiu *et al.* [60], [61], is a secret associated with a point in space and time, an adversary not at a specific place or time should be unable to produce a valid tag, making spoofing the location no longer possible. But this work involves an extra infrastructure to generate the location tags. A distinctive feature of our solution is that we rely on users to prove an occurred encounter by themselves without seeking the facilitation of any infrastructure.

SMILE [52] is similar to our model in its key-exchange encounter proof. In SMILE, encounter keys are broadcast to co-located users as shared state that can be used later to prove participation in a prior encounter. The primary difference between SMILE and our work is that SMILE is vulnerable to snooping attacks by local adversaries, since all the co-located users can see the keys; while our work provides strictly confidential peer-to-peer communications.

Secure communications require the establishment of cryptographic keys. The authors

in [55], [1] propose to take advantage of the correlated information only available to two users to generate a key, while a third eavesdropping user could not learn anything about the key. They showed that it is possible to create identical keys at two users with the assistance of a shared authenticated public channel. Nevertheless, the assumption of the authenticated channel even before key establishment starts is unrealistic. In our work, key extraction is performed without any authenticated channel. According to [25], device pairing based on out-of-band (OOB) channels can not provide a level of security that is as strong as is originally assumed.

We also take advantage of Attribute Based Encryption (ABE) [47], [79] algorithm for secure communication channel construction and verifiability. Only the users having exactly required attributes are able to decrypt a message from the initiator. Hence, privacy is strictly preserved to the exactly matched users. In the mean time, the matched users can automatically verify to the initiator that they indeed possess the required attributes.

Chapter3

Secure Friend Discovery Based on Encounter History

Before we present the details of our proposed scheme, a brief high level overview is introduced as follows:

1. Mobile devices automatically establish shared keys with encountered peers, and record the encounter information with the keys.

2. When a mobile owner is in another event, and wants to make friends with those who share encounters in certain prior events, he can activate the application.

3. The mobile device broadcasts those encrypted events to nearby peers. Only the users who share those encounters can decrypt the events.

4. When the number of matched encounters is larger than a threshold, the system recommends the two users as friends.

Next, we will give a detailed description of each step.

3.1 Algorithm Design

3.1.1 Encounter Key Establishment

In an event, users carrying a smartphone, laptop or other mobile devices can sense the presence of each other in the proximity by short-range wireless communications via Bluetooth/WiFi interfaces. We define the mutual detection as an encounter. During each encounter, two connected users establish a random symmetric key: an encounter key.

It is a big challenge to establish cryptographic keys in a dynamic mobile social network, where a certificate authority or a key management infrastructure is not always available. With peer-to-peer association being formed on-the-fly, it is necessary to establish keys

between wireless peers without resorting to a fixed infrastructure. In this work, we adopt the radio-telepathy protocol introduced in [53] by exploring the special properties of the wireless channel: the underlying channel response between two devices is unique and decorrelates rapidly with distance.

In typical multipath environments, the signal fading is location-specific and reciprocal, i.e., the fading is the same between two users, Alice and Bob, no matter it is Alice→Bob or Bob→Alice. And this fading decorrelates over distance of the order of half a wavelength, λ. An adversary, Eve, who is more than $\lambda/2$ away from both Alice and Bob, experiences fading channels to Alice and Bob that are statistically independent of the fading between Alice and Bob [53]. Thus, the signal fading is unique to the transmitter-receiver pair, which allows us to create shared encounter secret keys, even in the presence of an eavesdropper.

Let $\mathbf{h}$ be the magnitude of the system transfer function of the mutipath fading channel between Alice and Bob, and its value at a given time t is $h(t)$. Our goal is to estimate the parameter $\mathbf{h}$ with observed value $h(t)$. To achieve this, Alice and Bob need to transmit known probe signals to each other. After receiving the probe signals, they compute an estimate $\hat{h}$ of $\mathbf{h}$. Since practical radios are half duplex, the probe signal transition is conducted only one direction at a time (not simultaneously). Therefore, two successive probes are needed. And between these two successive probes, $h(t)$ changes slightly due to reciprocity, which is a fundamental property of electronic wave propagation [63]. The received signals at Alice and Bob can be written as

$$r_a(t_1) = s(t_1)h(t_1) + n_a(t_1) \tag{3.1}$$

$$r_b(t_2) = s(t_2)h(t_2) + n_b(t_2) \tag{3.2}$$

where t_1 and t_2 are the time instants at which the probe signals are received by Alice and Bob, respectively; s(t) is the known probe signal; and n_a and n_b are independent noises at

Alice and Bob, respectively. Then the estimates of $\mathbf{h}$ can be expressed by:

$$\hat{h}_a(t_1) = h(t_1) + z_a(t_1) \tag{3.3}$$

$$\hat{h}_b(t_2) = h(t_2) + z_b(t_2) \tag{3.4}$$

where z_a and z_b are the noise terms related to n_a and n_b after processing by the estimate function of $\mathbf{h}$. Since the noise terms at two parties are independent, and there is a slight time lag between two successive probes, the value of $\hat{h}_a(t_1)$ and $\hat{h}_b(t_2)$ tend to be unequal. However, they are highly correlated if the time lag $\tau = |t_1 - t_2|$ is small enough. This correlation unique to the transmitter-receiver pair provides the basis to establish an encounter key, and also to prevent eavesdroppers.

Assume a malicious user Eve can overhear the probe signals from Alice and Bob. Let h_{be} and h_{ae} be respectively the parameters of the channel for Bob→Eve and for Alice→Eve. The values of h_{be} and h_{ae} are completely different from $\hat{h}$ estimated by Alice and Bob when Eve is more than $\lambda/2$ away. For example, at 2.4 GHz, Eve being roughly 6.25 cm away would not get any useful information from eavesdropping.

Next, Alice and Bob need to use the estimate of $\mathbf{h}$ to get an identical bit-string suitable as a key. They repeatedly send the probe signals for n times and produce a sequence of estimates $H_a = \{\hat{h}_a^1, \hat{h}_a^2, ..., \hat{h}_a^n\}$ and $H_b = \{\hat{h}_b^1, \hat{h}_b^2, ..., \hat{h}_b^n\}$. Then, the estimated values are quantized through a quantizer Q:

$$Q(x) = \begin{cases} 1, x > q_+ \\ \\ 0, x < q_-, \end{cases} \tag{3.5}$$

where q_+ and q_- are reference levels determined by channel statistics.

Since the transfer function of the multipath fading channel is the same in two directions, the sequence of channel estimates at Alice and Bob are random variables drawn from the

same underlying probability distribution. To take advantage of this feature, Alice checks her channel estimates H_a and finds out the locations of excursions that have $\geq m$ successive estimates above q_+ or below q_-, where m is a system parameter. From all these excursions, Alice randomly selects a subset, and sends to Bob the indices of the channel estimates lying in the centers of the excursions. Assuming the selected subset contains k excursions, the message Alice sends to Bob is in the form of $L = \{l_1, l_2, ..., l_k\}$. Upon receiving the message, Bob then checks whether his estimate sequence H_b contains at least $m - 1$ successive estimates centered around each index in L, whose values are either above q_+ or below q_-. Bob places such good indexes into $\tilde{L}$, and sends it to Alice. Bob and Alice compute $Q(H_a)$ and $Q(H_b)$ with each index in $\tilde{L}$, and generate a sequence of bits. By carefully choosing the values of q_+, q_-, and m, the generated bit-strings at two parties are identical with a very high probability. And these bit-strings are statistically random, thus can be used as an encounter key. Figure 3.1 depicts the symmetric key establishment, where a, b, c, d are four mobile users; K_{ij} is the key shared between user i and user j; and a line between two users indicates that they share a common key.

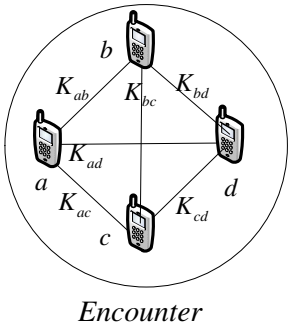


Figure 3.1: An illustration of symmetric encounter keys.

3.1.2 Event Selection

For each event, the mobile devices record the event information such as location, time, what it is about, and anything else the user wants to put to describe the event. We let L denote the location, T denote the time, and D denote the description. In this way, whenever

an event E is recorded, the mobile device stores a tuple about this event,

$$E :< L, T, D >, < K_1, K_2, ..., K_{n_E} > \tag{3.6}$$

where K_i $(i = 1, 2, ...n_E)$ is the encounter keys established for this event, and n_E is the number of encounter keys.

As the device owners attend more events, the devices obtain a sequence of events $\bar{E} = E_1, E_2, ..., E_k$. When a user is attending a new event, and wants to know others with similar background or interests, he can broadcast a friend making request, which includes selected events that are encrypted with the encounter keys. As an example, a computer scientist who is currently in an international conference on networking may want to know other attendees who have also attended INFOCOM 2013 and MobiCom 2012, denoted by E_p and E_q, respectively. In this case, he can select these two events and include their encrypted version in the request message. The goal of our system is to find out the people who have attended the same events, i.e., share common encounters with the initiator. By allowing the initiator to select events according to his preference for making friends, our system is designed to be customized.

3.1.3 Event Matching

When nearby users receive such a request with encrypted events, they are supposed to find the matched ones in their local records. Since the events are encrypted with encounter keys, only the users who did participate in the same events can decrypt them; thus our method successfully prevents the encounter information from being leaked to any server or to any user pretending to be in the same event.

Apparently, there are multiple encrypted versions for each event because there are multiple encounter keys established for that event, and there are multiple events for each user. If a user tries each of its keys to decrypt the events, it would be time- and energy-

consuming because decryption is computationally expensive. To speed up the retrieval process, we need to narrow down the retrieval scope. One obvious way is to narrow down the scope to a specific event first, then search the specific key within this event. Since the retrieval process is more lightweight than decryption, we believe it would be more suitable for resource constrained mobile devices. As described in (3.6), location L and time T are the most basic information about an event; thus we use $Hash(L, T)$ to represent the particular event, and use $Hash(K_i)$ to represent the key K_i. These hash values provide retrieval indices while concealing the sensitive information. Therefore the friend making request should have the following format:

$$\begin{aligned} Hash(L,T), \quad & < Hash(K_1), (E)_{K_1} >, \\ & < Hash(K_2), (E)_{K_2} >, \\ & \ldots \\ & < Hash(K_{n_E}), (E)_{K_{n_E}} > \end{aligned} \tag{3.7}$$

where $(E)_{K_i}$ is the encrypted event E with encounter key K_i.

Since each user has recorded multiple events with each event possessing multiple keys, we employ B+ trees to manage the encounter information for fast retrieval. For each user, an *event B+ tree* is constructed with each entry containing a hashed value $Hash(L, T)$ as the search key, the event (L, T, D), and a pointer to a *key B+ tree* storing all the encounter keys for this particular event. In a *key B+ tree*, the search key is the hashed encounter key $Hash(K_i)$ and the value is the shared key K_i. An example B+ tree is illustrated in Figure 3.2, where H_i is a hash value representing a search key (a hashed event in an *event B+ tree* or a hashed key in a *key B+ tree*), and we assume $H_i < H_{i+1}$ in this tree.

Whenever a peer receives a request message, it needs to answer the following two questions: which encounters the peers have, and which keys they share. Basically, it will perform the following two steps:

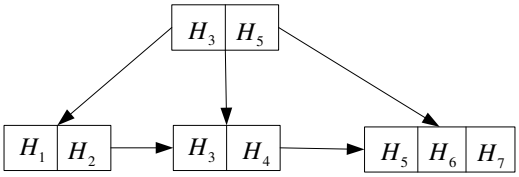


Figure 3.2: An example B+ tree.

1. it visits its *event B+ tree* based on $Hash(L, T)$ to pin-point the specific event.

2. it further visits *the key B+ tree* of this specific event to find out whether they share a common key.

Based on $Hash(K_i)$, the receiver can easily find the encounter key if it shares a common encounter with the initiator. Finally, it would be able to decrypt the event E, which could be some basic information about a true event (such as INFOCOM 2013), or some private message such as romantic request from the initiator to a specific receiver. Back to our previous example, when Kate receives a request message including two events E_p (INFOCOM 2013) and E_q (MobiCom 2012) from James, she first searches her *event B+ tree* for $Hash(L_p, T_p)$ and $Hash(L_q, T_q)$. If two matches are found, Kate knows that she did encounter the initiator in INFOCOM 2013 and MobiCom 2012. To further pin-point the shared keys established during the encounters, she searches her *key B+ trees* corresponding to the events INFOCOM 2013 and MobiCom 2012. If Kate did share such encounters with the initiator, she should be able to find K_p and K_q. Since all the communications are encrypted with encounter keys, they are completely confidential even in the presence of other peers.

For a b-order B+ tree, finding a record requires $O(\log_b^n)$ operations. Assume there are m events with each event having n encounter keys. Then the complexity of our event matching procedure is $O(\log_b^m) + O(\log_b^n)$ based on our model.

3.1.4 Matching Verification

Next, the receivers who can decrypt some of the events need to verify themselves to the initiator. Let's use an example to illustrate this procedure. To verify herself, the receiver Kate needs to send a message back to the initiator James, which includes $Hash(L_p, T_p)$ and $Hash(L_q, T_p)$ to verify the encounter events, and also $Hash(K_p)$ and $Hash(K_q)$ to verify the keys. Besides, to prove that she can indeed decrypt the events, Kate should include a *nonce* in plaintext, and use K_p and K_q to produce signatures for the *nonce*. Upon receiving the message sent back, James compares $Hash(L_p, T_p)$ and $Hash(L_q, T_p)$ with its local record of the broadcasted request. For each match found, he further determines the keys based on $Hash(K_p)$ and $Hash(K_q)$. Finally, he finds that K_p and K_q are used. With this information and the plaintext *nonce*, he computes the signatures $(nonce)_{K_p}$ and $(nonce)_{K_q}$, and compares them with the previously received signatures. If they match, Kate successfully verifies herself, indicating that James and Kate do have common encounters in events E_p and E_q, and their symmetric keys established during these two encounters are K_p and K_q. This process is illustrated in Figure 4.2, where ($\hat{E}_a$, ..., $\hat{E}_p$, $\hat{E}_q$, ...) are selected events in the format of (3.7).

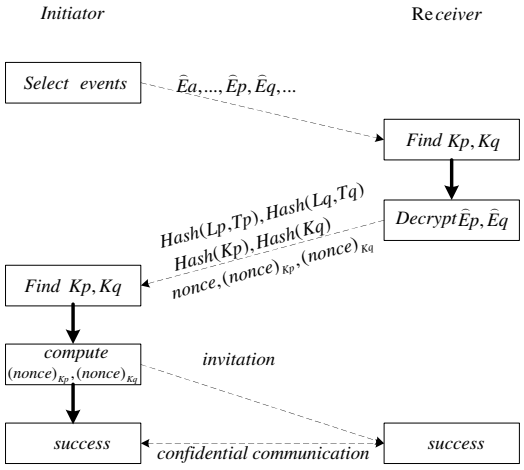


Figure 3.3: Matching process.

After all the receivers who can decrypt some of the events sending their verification

messages back to the initiator (for those who do not share any common encounters, no message needs to be sent back), the initiator decides whether or not to make friends with them. If yes, an invitation is sent out. By default, the system invites the peers whose numbers of common encounters are larger than a threshold.

3.2 An Alternative Design

In this section, we present an alternative to our peer-to-peer friend discovery protocol to provide a more complete perspective on the design space. Instead of establishing symmetric keys between every encounter pairs, we propose to establish a group key among all the encounter users. The mobile devices are becoming more capable and sophisticated, especially the smartphones, which have sensitive sensors embedded. These sensors can collect environment readings such as temperature, humidity, and light, in logical positions, which are random and different at different logical locations. The crucial insight that the encountered users are located in the same environment provides the basis for group key generation. Although the measurements of an environment on different devices are not guaranteed to be identical, they are quite close. A straightforward method to translate the close values to identical bit-strings is to adopt some fault-tolerant algorithms or quantizer. We believe this is a possible way to obtain a group key. In this section we merely provide a direction here, while the details are reserved for future research.

This alternative design provides storage and computation efficiency at the cost of weakened privacy guarantees. Comparing with the peer-to-peer design, the number of encounter keys is significantly reduced, since all the engaged peers of one event share a common group key, instead of sharing one key with each of the encountered peers. As the events are encrypted by the same group key in the event matching phase, the computation of encryption and decryption is less complicated. However, the primary drawback of this design is that all the encounter peers of the same event can decrypt the massage. When the peer-to-peer confidential communication is not guaranteed, the privacy protection

is weakened. Therefore this alternative design yields a tradeoff between efficiency and privacy.

3.3 Performance Evaluation

We conduct experiments with two Android phones. One is equipped with 1GB RAM and a dual-core 1.2 GHz Cortex-A9 processor (Galaxy S2) announced in 2011; the other is equipped with 2 GB RAM and a dual-core 1.5 GHz processor (Galaxy S3) announced in 2012. We use Bluetooth for message exchange. The basic structure of our scheme can be broken down into two steps: (1) encounter key establishment; and (2) friend discovery based on encounter history. In this section, we examine the feasibility of these two steps.

3.3.1 Encounter Key Establishment

Since the key generation algorithm adopted in this work is introduced by Mathur *et al.* [53], which provides sufficient theoretical support and experimental analysis, we will not duplicate the work here to prove its feasibility. However, we still need to point out the following two important features of the algorithm about key generation: the randomness and the key generation rate.

1) *Randomness*: Guaranteeing that the generated keys are random is crucial, since non-random behavior in bit sequences can be exploited by an adversary to reduce the time-complexity of cracking the key. According to Mathur *et al.* [53], the proposer of this key extraction mechanism, the key generation algorithm in [53] provides random bits. A widely accepted benchmark for testing randomness is Maurer's universal statistical test [54], which was adopted by [53] to evaluate the randomness of the bit sequences. The results indicate that the average entropy of the bit sequences is very close to the value expected for a truly random sequence.

2) *Generation Rate of Secret Bits*: The generation rate is affected by several factors, such as the parameters q_+, q_-, m, the rate f_s at which Alice and Bob probe the channel

between them, and the rate at which the channel varies, represented by the maximum Doppler frequency f_d. By carefully tuning the values of these parameters, the desired generation rate can be achieved. The impact of each parameter has been thoroughly studied in [53]. One noteworthy thing is that there exists a tradeoff between probability of error and rate of secret bit generation.

3.3.2 Friend Discovery Based on Encounter History

Friend discovery based on encounter history is the focus of this paper; we thus mainly focus on evaluating the performance of this step. And the analysis and evaluation are carried out from the following aspects: 1) memory storage; 2) communication overhead; 3) energy consumption.

1) *Memory storage*: Encounter history recorded in a node takes the major fraction of its memory. We assume that there are R stored events, which means that there are R vertices in the *event B+ tree* of the node with each vertex representing one event E having the following information

$$Hash(L,T), < L, T, D >, \mathbb{P}$$

where $\mathbb{P}$ is a pointer to the *key B+ tree* of this particular event. The *key B+ tree* consists of n_E encounter keys. And each vertex in the *key B+ tree* is in the form of

$$Hash(K_i), K_i$$

We use S_H and S_K to denote the lengths of the hash value and the encounter key, respectively. For the event information $< L, T, D >$, the size is denoted by S_I. Apparently, the size of one event record is equal to the size of one vertex in *event B+ tree* plus the size of the whole *key B+ tree*. Thus, without considering the space for a pointer, the estimated memory for one event is $S_H + S_I + n_E(S_H + S_K)$. Then the total memory cost for the

encounter history $[E_1, E_2, ..., E_i, ..., E_R]$ recorded in a node is

$$\sum_{i=1}^{R}\{S_I + n_{E_i} \cdot S_K + S_H(1 + n_{E_i})\} \tag{3.8}$$

For event information, we assume that the size S_I is limited to 300 bytes, which should be enough to provide all necessary information for an event. We further assume a key length S_K of 256 bits (32 bytes) by following the general requirement about symmetric key security. The length of hash values S_H is assumed to be 256 bits (32 bytes) too. In order to provide more intuitive results, Figure 3.4 plots the relationship between memory storage and the average number of keys for one event when R varies. One can see that the memory storage increases with the increase of the average number of keys, which indicates that a user encounters more people in an event. Besides, when the number of recorded events increases, the memory cost increases as well.

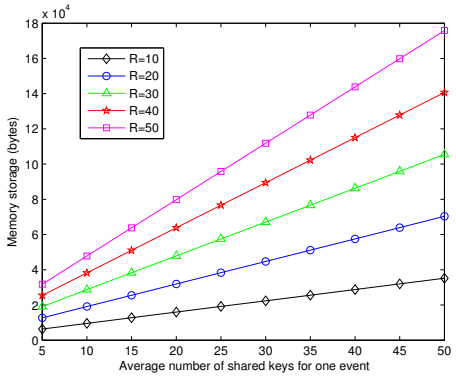


Figure 3.4: Memory storage.

2) *Communication overhead*: The friend discovery process involves some message exchanges, such as matching event broadcasts and matching verifications. As illustrated in Figure 4.2, there are two steps before an invitation is sent. These two steps contribute the most to the communication overhead. First, the initiator broadcasts a request message with Z selected events with each being in the form of (3.7). We can see that there are $1 + n_E$ hash values and n_E encrypted versions for one event. Thus the storage size for an event

is $S_H(1 + n_E) + S_I \cdot n_E$. Assuming that the initiator selects Z events for matching. The estimated message size of the whole broadcast request is

$$\sum_{i=1}^{Z} \{S_H(1 + n_{E_i}) + S_I \cdot n_{E_i}\} \tag{3.9}$$

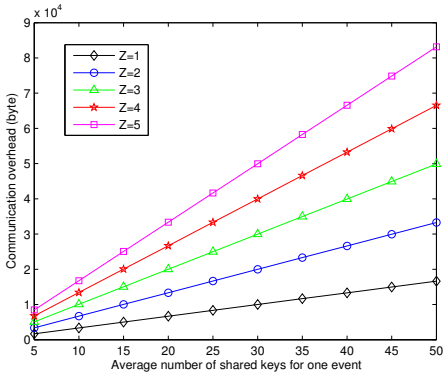


Figure 3.5: Communication overhead of an initiator.

The second step is matching verification. To verify an event, a receiver needs to send back $Hash(L, T)$, $Hash(Key)$, $nonce$, and $(nonce)_{key}$, which contribute $2S_H + S_{nonce} + S'_{nonce}$ to memory storage. Here, we assume the size of nonce S_{nonce} to be 32 bytes, and its encryption version S'_{nonce} is also of the size of 32 bytes. If a receiver has M matching events, the total communication overhead is

$$M(2S_H + S_{nonce} + S'_{nonce}) \tag{3.10}$$

Note that the communication overheads are different for an initiator and a receiver, expressed by (3.9) and (3.10), respectively. The quantitative relationship between communication overhead and the number of encounter keys n_E with different Z is illustrated in Equ. (3.9). We observe that the communication overhead of the initiator increases with the number of encounter keys and the number of selected events. And the communication overhead of a receiver increases with the number of matching events, as

confirmed by Equ. (3.10).

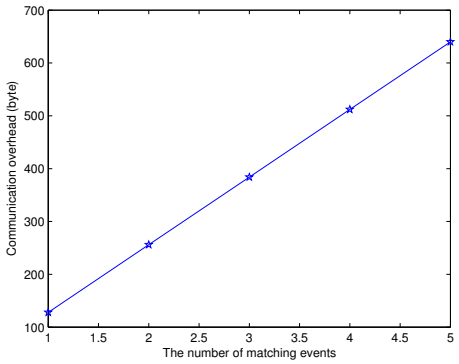


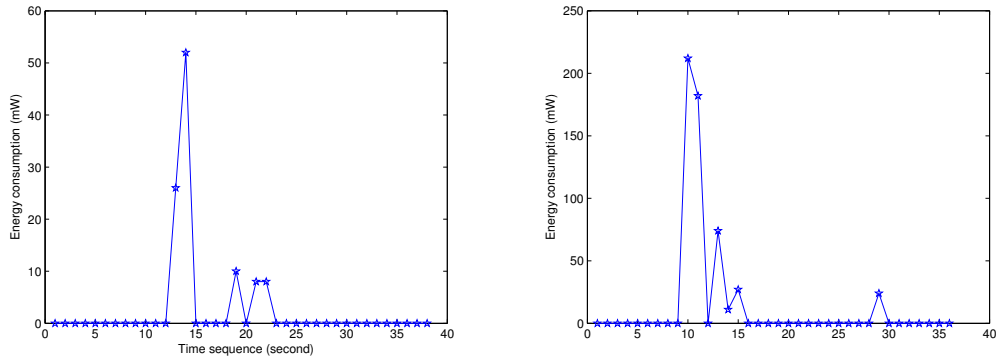
Figure 3.6: Communication overhead of a receiver.

3) *Energy Consumption*: Energy consumption is an important metric to assess the application performance for mobile devices. In our scheme, the encryption/decryption algorithm and message transmission are the main causes for energy consumption. Thus we focus on the process of sending a message after encrypting it, and study the impact of message size and transmission distance. Figure 3.7 shows that the energy consumption variates over time with the transmission distance being 1m, and the sizes of messages being 25KB and 100KB. One can see that with a larger message size, the energy consumption during the transmission is much larger. Besides, based on the non-zero values we observe that both can complete message transmission within 10 seconds.

Similarly, Figure 3.8 plots the energy consumption with transmission distance being 5m. By comparing Figures 3.7(a) vs. 3.8(a) and Figures 3.7(b) vs. 3.8(b), we conclude that even with the same message size, a larger distance results in a large energy consumption. We believe that this is because the sender consumes more power to send the signal to make sure it reaches the destination further away.

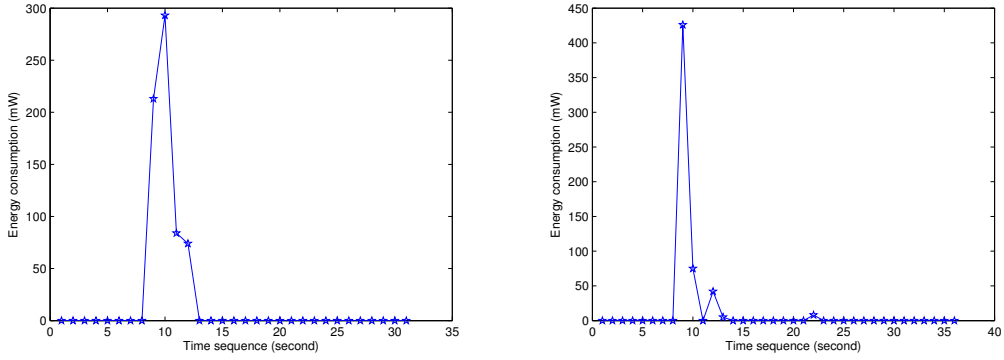
3.4 Conclusion

While mobile social networking enables a large number of exciting applications, it also arouses serious privacy concerns. In this work, we propose a novel idea of friend



(a) Message size=25KB (b) Message size=100KB

Figure 3.7: Energy consumption with different message sizes when distance $= 1m$.



(a) Message size=25KB (b) Message size=100KB

Figure 3.8: Energy consumption with different message sizes when distance $= 5m$.

discovery based on encounter history in a privacy-preserving way. By exploring reciprocity property of electromagnetic wave propagation, our algorithm allows two encounter users to agree upon a symmetric cryptographic key without any keying material pre-distribution or initialization. We also guarantee the location privacy and encounter privacy without relying on any trusted centralized server or any existing trusted social relationships. Considering the dynamic property of mobile social networks, we believe our ad-hoc design is more suitable and general. According to the performance evaluation and theoretical analysis, we claim that our scheme is effective and feasible.

In our future research, we intend to increase the users' flexibility of presence-share by tuning their access controls to their social environments. In other words, we will consider the case when users can control who can share the encounter information among the participants.

Chapter4

Customized Privacy Preserving Friend Discovery

4.1 Protocol Overview

Since a user may want to find a friend according to his/her specific preference, a profile in this work is designed in a more detailed way. Each attribute of the profile has two fields: an attribute header and an attribute value, so that a user can set a customized request profile by choosing certain attribute headers and giving each attribute a specific value. An example profile is given in Table 4.1. Note that all of the selected attributes in the request profile are required to be owned by a matched user. However, it often forms a deadlock that the initiator wants to find a matched user who has all the designated attributes but is unwilling to disclose its request profile to other users.

In order to break this deadlock, we propose to adopt Attribute Based Encryption to encrypt the request and construct a secure communication channel. Only the matched users who have the same required attributes are able to communicate with the initiator. However, this raises another challenge: when a receiver gets a request encrypted with ABE, how does it know whether it owns the required attributes? If the receiver tries every possible subset of its attributes to do the decryption, it would be computationally prohibitive. Particularly, most users would be unmatched users who would come to the conclusion that they do not match the initiator after all the decryption attempts. This incurs significantly unnecessary computational overhead. Motivated by this fact, we design a filter tool to exclude most unmatched users before they conduct decryption.

Bloom filter [41,44,72] is an effective tool used to test whether an element is a member of a set. In this work, Bloom filter provides some hint that could be used by the receivers to figure out whether he/she has such required attributes. Since false positive matches are possible in Bloom filter, the candidate users who pass the Bloom filter test would need to

decrypt the request. If they can decrypt the request, they successfully prove that they do have the required attributes, which is termed as *verifiable* in this work. Then the initiator and the matched users can securely communicate with each other. For those users who do not pass the Bloom filter test, it is guaranteed that they are unmatched users and they do not need to do the decryption. Since the decryption operation is much more costly than hash functions, this Bloom filter approach efficiently decreases the computational overhead.

Table 4.1: An example profile

Attribute Header	Attribute Value
$Name$	Charlie
Age	28
$Interest$	tennis, video games, basketball
$Groups$	IEEE society, computing club
$Profession$	computer engineer
$Hometown$	Virginia
$University$	MIT

Next, we give a detailed description of each step.

4.2 Step 1: Secure Customized Profile Matching

4.2.1 Basic Matching Mechanism

The n attributes of an initiator form a profile set denoted by $S = \{a_1, a_2, ..., a_i, ..., a_n\}$. The empty Bloom filter is an array of m bits, with each bit set to 0 initially. There are k independent hash functions $h_1, h_2, ..., h_k$. Each h_i maps an attribute in S uniformly to one of the m array positions. To construct a Bloom filter for an attribute profile, we feed each element to all the k hash functions to obtain the corresponding positions in the bit array. These bits are then set to 1. Figure 4.1 shows the generation of a Bloom filter for the profile in Table 4.1 with $k = 3$. Note that $h_3(a_1)$ and $h_2(a_2)$ map to the same bit.

To query whether an element a' belongs to the set S, feed it to each of the k hash functions, and the corresponding $h_i(a')$ bits are checked. If any of the bits are 0, a' is definitely not in the set. This is because all the bits would have been set to 1 if it belongs to

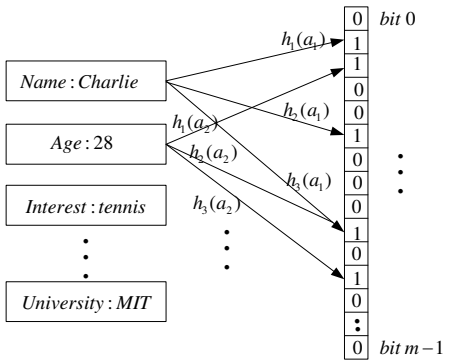


Figure 4.1: An illustration of Bloom filter with $k = 3$.

the set. If all of the bits are 1, the element is considered in the set. However, Bloom filter yields *false positive*, where $a' \notin S$ but its hashed positions have collectively been set to 1 by elements in S.

Assume that the hash function randomly selects each of the m positions with a uniform probability, the probability that a certain bit is not set to 1 after k hash functions is $(1-\frac{1}{m})^k$. After all the n elements are hashed and their bits are marked, the probability that the bit is still 0 is $(1-\frac{1}{m})^{kn} \approx e^{-\frac{kn}{m}}$. Therefore, the probability that it is 1 is $1-(1-\frac{1}{m})^{kn}$. The probability that k bits of an element are already marked, which would cause a false positive, is given by $(1-(1-\frac{1}{m})^{kn})^k \approx (1-e^{-\frac{kn}{m}})^k$.

To protect the profile privacy, the initiator would send a friend making request, which includes the Bloom filter of required attributes and other information encrypted with ABE:

$$\mathcal{B}, ABE.E(K_e||M||nonce)$$

The information encrypted by ABE has three components: K_e, M, and a nonce. K_e is a symmetric key used to construct a secure communication channel between two users. If a receiver can decrypt the ABE encryption, it can get the symmetric key and use it to encrypt the following communications. M is the specific message the initiator sends out for making friends, *e.g.*, "Hi, how are you?". The $nonce$ is used to verify the receiver,

which will be returned by the receiver to indicate that he/she can indeed decrypt the ABE. The returned message of a matched user is in the following format:

$$E_{K_e}(M'||nonce)$$

where M' is the response to M, *e.g.*, "I'm good."

Upon receiving such a request, a receiver first checks whether his/her Bloom filter includes all the marked bits in $\mathcal{B}$. If it does, this receiver is a candidate matched user; otherwise, it is not possible to match the requested attributes and the receiver will be excluded immediately. During this fast Bloom filter matching, unmatched users are filtered out without conducting the ABE decryption operations. Unnecessary computation and communication overhead is significantly reduced. On the other hand, the candidate matched users further decrypt the ABE by using his/her attributes corresponding to the marked bits in $\mathcal{B}$. Only those matched users who have the exact required attributes can successfully decrypt the ABE. Then the decrypted $nonce$ is returned back to the initiator for matching verification. And the following communication will be encrypted by the symmetric key K_e. An illustration of this procedure is shown in Figure 4.2. Even though there might be false positives in Bloom filter matching, those candidates who pass the Bloom filter matching but do not have the required attributes are not able to decrypt the ABE correctly, thus cannot verify to the initiator with the $nonce$.

We try to protect the profile privacy by hiding it via Bloom filter and ABE. People may argue that a Bloom filter reveals privacy since a malicious user can launch a dictionary attack. A successful dictionary attack requires the attacker to obtain all possible attributes and their corresponding values. Since we let the initiator set the request profile according to his/her specific preference, the chosen attributes and the attribute values are completely up to the user. The space of possible attributes and their values are mostly very large, which makes the dictionary attack infeasible. Therefore we believe it would be extremely difficult

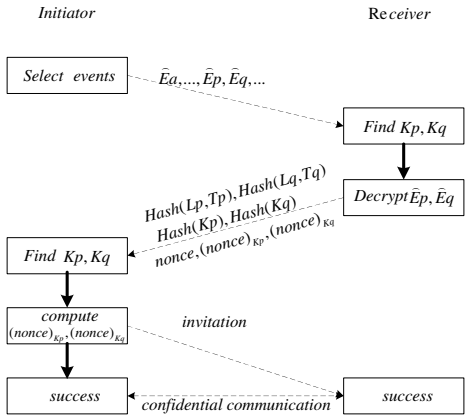


Figure 4.2: Matching procedure.

to launch such an attack by guessing the request profile correctly. Note that each request in our work has a valid time; thus a response to an expired request will be simply dropped.

4.2.2 Collusion Resistant Matching Mechanism

While most existing work neglects collusion attacks in privacy preserving matching, we take a deep investigation on it. One of the most popular private attribute matching mechanisms is the private set intersection (PSI). The focus of PSI is to ensure that as little information as possible is disclosed beyond the matching results. Nevertheless, the intersection attributes are disclosed, even if the number of intersections is not larger than a threshold. In this case, the two parties will not be friends but they do get a piece of private information about each other.

Next, we show how this could lead to collusion attacks with an example. We assume that Alice is the initiator of a friend making request, which requests at least 6 matching attributes. When Bob receives the request and detects that he has 4 matching attributes, he will not be able to make friends with Alice. However, he obtains the 4 matching attributes. And the same thing happens to Charlie who has 5 matching attributes. Apparently, both of them are not eligible to make friends with Alice; thus they are not supposed to know the attributes of Alice. However, there is a good chance that if they put their matching attributes together, they could figure out what the complete set of private attributes is. This

problem exists in most related work, which is also identified by [85]. To the best of our knowledge, no sound technical solution was proposed to counter such attacks.

In our mechanism, a similar problem exists in the Bloom filter matching procedure. Again, we use the aforementioned example. Assume Bob receives Alice's request and finds out his Bloom filter includes part of the marked bits but not all of them. Apparently Bob is not a matched user, but he can figure out the attributes of the overlapping marked bits. Let's say Charlie has some of the request attributes too. If Bob and Charlie collude with each other, they may be able to guess Alice's request attributes successfully.

To tackle the collusion attacks, we conceal the Bloom filter by hashing it. The resulted hash value $\mathcal{H}$ is included in the request for profile matching instead of the Bloom filter $\mathcal{B}$ itself. Figure 4.3 shows the generation of $\mathcal{H}$.

Since $\mathcal{H}$ only contains some random numbers, unmatched users definitely get nothing. On the other hand, this also increases the difficulty for matched users as they can get the exact $\mathcal{H}$ only by hashing the exactly same matched attributes' Bloom filter. However, they have no clue about which attributes are required. So we let the initiator reveal the required attributes' headers, such as Name, Age, Interests, *etc*. Then the receivers can pull up the corresponding attributes and map them with their values to a Bloom filter to further get $\mathcal{H}'$. If $\mathcal{H}' = \mathcal{H}$, the receiver is a matched user; otherwise it is not.

Apparently, only the users with exactly the same matching attributes satisfy $\mathcal{H}' = \mathcal{H}$. And the unmatched users cannot get any information about the profile privacy based on the random values in $\mathcal{H}$. Even if they have part of the required attributes, their resulted $\mathcal{H}'$ would be completely different from $\mathcal{H}$. Of course, the revealed attribute headers leak some information, which indicates what aspects (Name, Age, Interests) the initiator is interested in. But we would argue that such information without specific values is not that sensitive.

To make it more rigorous, we measure the privacy leakage level by introducing the concept of *entropy*. Following the definitions in [85], we define *attribute entropy* and *profile*

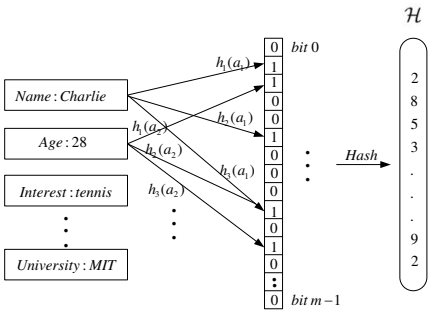


Figure 4.3: The generation of $\mathcal{H}$.

entropy as follows:

Definition 1 (Attribute Entropy): For an attribute a_i with v_i values $x_1, x_2, ..., x_{v_i}$, $P(a_i = x_j)$ is the probability that the attribute a_i takes the value x_j. The entropy of the attribute a_i is defined to be

$$e(a_i) = -\sum_{j=1}^{v_i} P(a_i = x_j) \log P(a_i = x_j) \tag{4.1}$$

Definition 2 (Profile Entropy): The entropy of a profile S is defined to be

$$e(S) = \sum_{i=1}^{n} e(a_i) \tag{4.2}$$

The larger the entropy, the stronger the privacy preservation. A user can always come up a request which satisfies $e(S) \geq \theta$, where θ is the threshold of an appropriate privacy level set by the initiator according to his/her customizable privacy protection requirements

.

To recognize the same attribute with different formats is a very challenging problem. Even a small change in the input of a hash function produces a very different output. For example, the hash function results of “basketball” and “Basketball” are completely different. So we need to normalize the profile attributes before mapping them to a Bloom filter to make sure that the attributes that are supposed to be equivalent to each other yield

the same hash result. The simple case is the inconsistency caused by typing such as letter case, punctuation, spacing, *etc*. A more complicated case is the inconsistency caused by semantic equivalence. There are some well studied techniques for word normalization [73] in research areas such as search engines and corpus management [28], which can perfectly fit in our mechanism. To be more focused on the core techniques of friend discovery, we ignore the impact of word normalization in our analysis.

After the filtering phase, most unmatched users are excluded. For the users that pass the Bloom filer, there is a big chance that they do satisfy the required attributes of the initiator, but there is still a chance that they pass the filter with false positives.

4.3 Step 2: Secure Channel Construction

To further eliminate false positives, we take advantage of the Attribute Based Encryption (ABE) algorithm for secure communication channel construction and verifiability. In the following we introduce the details of adopting ABE in our protocol by following the fundamental work of Sahai *et al.* [69].

Let $\mathbb{G}_1$ be a bilinear group of prime order p, and let g be a generator of $\mathbb{G}_1$. Additionally, let e: $\mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ denote a bilinear map. A security parameter κ, determines the size of the group.

We also define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$, and a set S of elements in $\mathbb{Z}_p$:

$$\Delta_{i,S} = - \sum_{j \in S, j \neq i} \frac{x - j}{i - j} \tag{4.3}$$

Applying ABE in our protocol needs to perform the following four steps: Setup, Key Generation, Encryption, and Decryption, which are detailed as follows.

Setup

Choose $t_1, ..., t_{|S|}$ uniformly at random from $\mathbb{Z}_p$. Then choose y uniformly at random

from $\mathbb{Z}_p$. The published public parameters are:

$$T_1 = g^{t_1}, ..., T_{|S|} = g^{t_{|S|}}, Y = e(g,g)^y.$$

and the master key is

$$t_1, ..., t_{|S|}, y.$$

Key Generation

An $n-1$ degree polynomial q is randomly chosen such that $q(0) = y$. The private key consists of components, $(D_i)_{i \in S}$, where $D_i = g^{\frac{q(i)}{t_i}}$ for every $i \in S$.

Encryption

First, a random value $s \in \mathbb{Z}_p$ is chosen. Then the whole message is encrypted by $\tilde{M} = (K_e||M||nonce)$. The ciphertext is then published as

$$E = (E' = \tilde{M}Y^s, \{E_i = T_i^s\}_{i \in S}).$$

Decryption

When a candidate matched user receives such an encrypted message E, he/she uses the attribute set obtained from Bloom filter matching to decrypt it. The ciphertext can be decrypted as:

$$\begin{aligned}
&E' / \prod_{i \in S'} (e(D_i, E_i))^{\Delta_{i,S(0)}} \\
&= Me(g,g)^{sy} / \prod_{i \in S'} (e(g^{\frac{q(i)}{t_i}}, g^{st_i}))^{\Delta_{i,S(0)}} \\
&= Me(g,g)^{sy} / \prod_{i \in S'} (e(g,g)^{sq(i)})^{\Delta_{i,S(0)}} \\
&= M
\end{aligned}$$

If the receiver is an exactly matched user, then we have $S = S'$. Thus, the above equation holds. The last equality is derived from using polynomial interpolation in the exponents. Since the polynomial $sq(x)$ is of degree $d-1$, it can be interpolated using d points [69].

4.4 An Alternative Design

An alternative method to further eliminate false positive probability is to apply more than one Bloom filter. While Attribute Based Encryption is relatively expensive in computational overhead, Bloom filters are more efficient. Our previously introduced mechanism applies Bloom filter once to filter most of the unmatched users. Now we apply another Bloom filter in exactly the same way to the users that have passed the first Bloom filter. Apparently, the false positive rate could be further reduced with less overhead since the second Bloom filter is only applied to a small fraction of the users.

The k hash functions of the second Bloom filter should be different from those of the first one, such that the two Bloom filters are independent with each other. The false positive probability of one Bloom filter is given by $(1-(1-\frac{1}{m})^{kn})^k \approx (1-e^{-\frac{kn}{m}})^k$ as we have analyzed before. Only users passing both Bloom filters are considered potential friends. After two Bloom filters, the false positive probability would be exponentially reduced to $(1-e^{-\frac{kn}{m}})^{2k}$. Of course, more Bloom filters can be introduced if necessary depending on the application requirements of the false positive probability.

From the perspective of profile matching, multiple Bloom filters can achieve privacy preserving, collusion resistance, and sufficient accuracy. Comparing with existing work that either relies on PSI (PSI-CA) or resorts to homomorphic cryptography, this method significantly reduces computation and communication overhead.

In this work, we take one step further to consider the secure channel construction and verifiability, which makes the adoption of Attribute Based Encryption necessary. Even though it is more expensive, considering that most unmatched users have been filtered

out and only a small fraction of potential users conduct ABE, we believe this overhead is acceptable.

4.5 Security Analysis and Performance Evaluation

4.5.1 Security Analysis

4.5.1.1 Profile Privacy

In our scheme, only users with sufficient matching attributes can decrypt the initiator's request and establish a secure communication channel with the initiator. In the basic mechanism of secure profile matching, the completely unmatched users with no overlapping attributes with the initiator would obtain no useful information from the request. All the information included in the request is a Bloom filter of the required attributes and other information encrypted by Attribute Based Encryption (ABE).

- For completely unmatched users, the Bloom filter is just some random 0/1 stream from which nothing can be inferred. And the request information is kept secret.
- For partially matched users, the Bloom filter leaks the overlapping attributes to the receivers to some extent, since they can reversely obtain the common attributes from the common marked bits in the Bloom filter. However, this problem exists in most of the existing work, such as private set intersection (PSI) and private set intersection cardinality (PSI-CA). The ABE encrypted information is kept completely secret.
- For exactly matched users, the profile privacy is not an issue, since they are eligible to be the initiator's friends. And the ABE encrypted information can be decrypted, so that the symmetric key K_e can be used to construct a secure communication channel to ensure that the following message exchange is secure.

To protect the profile privacy from the partially matched users and thwart the collusion attacks among them, we propose an enhanced mechanism: *collusion resistant secure profile*

matching. This mechanism guarantees that the profile privacy is obtainable only to the exactly matched user. By revealing the attribute headers of the required profile, the exactly matched users are able to directly find out the matching attributes. Although the attribute headers are revealed for efficient matching search, the values are kept secret. We believe that the space of the possible values are mostly large. It would be difficult to successfully launch dictionary attacks within a predefined time. Moreover, the initiator can define his/her privacy protection level in terms of entropy by customizing the required profile.

Figure 4.4 shows the entropy values with different attribute value spaces and different numbers of attributes. The spaces of different attributes are certainly different. To get an intuitive impression of this relationship, we use the average values for entropy calculation. Assume there are u values in average for each attribute a_i, with each value being taken at the same probability, the entropy of attribute a_i is $e(a_i) = \log u$. Since there are n attributes in the profile, the profile entropy is $e(S) = n \cdot \log u = \log u^n$. As shown in the figure, with a larger attribute value space, the entropy increases. Also, the entropy increases when the profile includes more attributes.

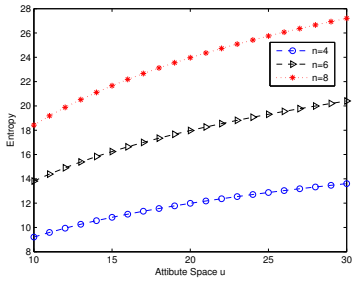


Figure 4.4: Entropy values with different attribute spaces.

4.5.1.2 Communication Security

In our protocol, a symmetric key is protected by the Attribute Based Encryption (ABE). Only the users with exactly the same matching attributes can decrypt the ABE and obtain the symmetric key for secure channel establishment. The following communication will be encrypted by the symmetric key. Any other unmatched users are not able to access the

key. Therefore our protocol realizes secure communications between the initiator and the matched users.

4.5.1.3 Verifiability

In majority of existing profile matching approaches, only one party learns the true result, and then he/she tells the other party. There lacks an efficient way for the other party to verify the result [85]. Our protocol is verifiable for both parties. In our protocol, we adopt Attribute Based Encryption to encrypt a symmetric key which is used in the following communication. Only matched users are able to obtain this key, and use it to encrypt the $nounce$ to the initiator. This automatically verifies to both parties that they share the required common attributes.

4.5.2 Performance Analysis

We use Bloom filter to efficiently filter out the unmatched users in our protocol in order to reduce unnecessary computational overhead. The price paid for this efficiency is that a Bloom filter yields false positives. It's a nice property that the false positive probability can be controlled by adjusting the filter size m and the number of hash functions k. Figure 4.5 shows that the false positive rate is a function of the filter size m and the number of hash functions k. We assume that the number of attributes in the profile is 6. In Tencent Weibo, the average number of attributes of each user is about 6. Apparently, as shown in the figure, a larger filter results in a smaller false positive rate. Also, we can see that when the filter size is close to 40, the false positive rate is below 0.04. Imagine we use two Bloom filters, the false positive rate would be $(0.04)^2 = 0.16\%$, which is extremely small.

In this work, we assume the filter size is 40. Given $n = 6$ and $m = 40$, the optimal value of k can be calculated by $(m/n)ln2$ [41]. Accordingly, the number of hash functions k is about 4.

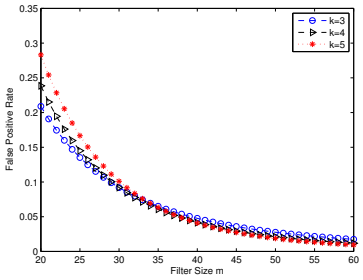


Figure 4.5: False positive rates with different filter sizes.

4.5.2.1 Computational Cost

In this subsection, we present a quantitative performance study of ABE in terms of computational cost, and compare it with the existing work FC10 (PSI) [9] and FNP (PSI-CA under HBC model) [17]. These work all falls into the same category of profile matching by treating the profile as a set of attributes and conducting attribute matching rather than measuring social proximity with dot product.

The computational cost is evaluated using the number of modular multiplications, exponentiation operations, and pairing operations. We only count the number of online operations. The details of the functions and operations of ABE are presented in Table 4.2. The comparison with PSI and PSI-CA is summarized in Table 4.3, where n and n' denote the number of attributes of the initiator and the receiver, respectively, mul_1, exp_1, exp_2 denote a 1024-bit multiplications, a 1024-bit exponentiations, and a 2048-bit exponentiations, respectively, and U is the number of participated users.

Table 4.2: Computational cost of ABE

Key Generation	Encryption	Decryption
$nexp_1$	$nexp_1 + 1$ pairing	n pairing

4.5.2.2 Execution Time

We assume the offline computation is performed in preset and is not counted into the protocol execution time.

Table 4.3: Computational cost comparison

Scheme	Cost
FC10 (PSI) [9]	$1.5nUmul_1 + (n+n')exp_1$
FNP (PSI-CA) [17]	$(2n+n'U)exp_2 + n'log(logn)exp_2$
ABE	$2nexp_1 + (n+1)\ pairing$

Since there exist optimization algorithms, we use existing benchmark test results instead. The results are tested on LG-970 smartphones, which has a iGHz-A8 processor, 512 MB RAM, Android v2.2 Operating System, and a 802.11 b/g/n WiFi interface. Table 4.4 summarizes the computational cost in terms of the execution time (ms) of different operations based on the Stanford Pairing-Based Cryptography (PBC) library [51], which in an open source C library that performs mathematical operations underlying pairing-based crypto systems.

Table 4.4: Execution time of different operations (ms)

Operation	mul_1	exp_1	exp_2	$pairing$
time	0.73	81.08	159.06	11

We quantify the computational cost under typical parameter settings $n = 10, n' = 10, U = 10$. Figure 4.6 shows the running time of three algorithms with the number of initiator's attributes n changing. Similarly, Figures 4.7 and 4.8 illustrate the running time with the number of receiver's attributes n' and the number of users U changing.

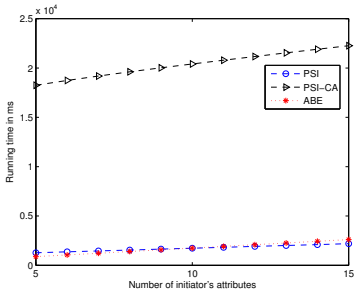


Figure 4.6: Running time with n changing.

From the figures, one can see that the computational cost of ABE is significantly smaller than that of PSI-CA, since PSI-CA uses exp_2 which would need more time. Another

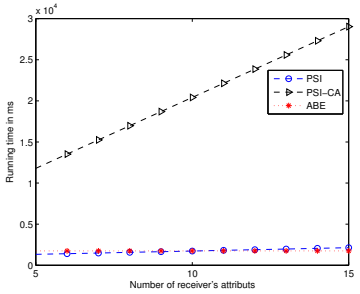


Figure 4.7: Running time with n' changing.

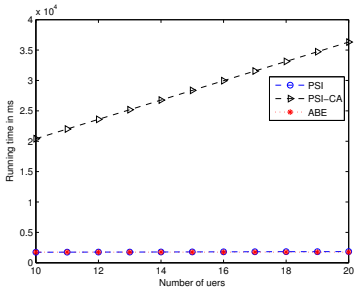


Figure 4.8: Running time with U changing.

observation is that the cost of ABE is similar to that of PSI. Noticing that our protocol only requires the receivers that have passed the Bloom filtering to conduct ABE while PSI requires every receiver to compute the set intersection, our protocol is much more efficient than PSI.

4.5.2.3 Communication Cost

Communication cost is evaluated in terms of the total number of message bits that are exchanged in the profile matching process. In our protocol, we assume that the Bloom filter size $m = 40$ bits, the symmetric key K_e is 128 bits, the messages M and M' are both 500 bits, and the $nonce$ is 32 bits. Messages after Attribute Based Encryption are at most 1024 bits. So in the basic mechanism, the message size of the initiator is 40+1024=1064 bits; the message size of the receiver is 500+32=532 bits. In the enhanced mechanism, the Bloom filter is replaced by a hash value, which is 128 bits, while other parts remain the same. The size of message exchanges can be derived similarly. In addition, the enhanced mechanism

needs to reveal the required attributes' headers as we explained before. Assuming that the longest attribute header is 20 letters, then the size of each header is 160 bits.

By summing up the message bits from both the initiator and a receiver, we can summarize that the overall communication cost in Table 4.5, where $t = 4$ and $q = 1024$ in the existing work.

Table 4.5: Communication cost comparison

Scheme	Cost
FC10 (PSI) [9]	$nn'U + tU(8n + 2n' + 12nt)$
FNP (PSI-CA) [17]	$2(n + n'U)q$
Basic Mechanism	1596
Enhanced Mechanism	$1684 + 160n$

Under a typical parameter setting $n = 10, n' = 10, U = 10$, we illustrate the communication cost in Figures 4.9, 4.10, and 4.11.

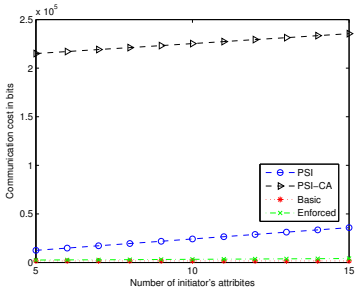


Figure 4.9: Communication cost with n changing.

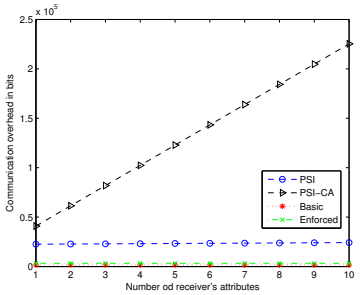


Figure 4.10: Communication cost with n' changing.

First, we can see that our basic mechanism has the nice property of fixed communication cost that is independent of the number of attributes or the number of users. Besides, the

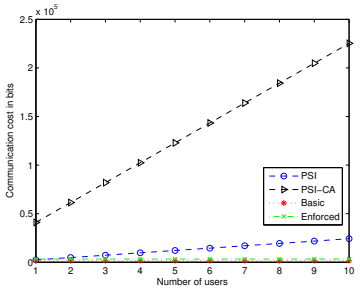


Figure 4.11: Communication cost with U changing.

communication cost is significantly lower than the existing work in three different cases. Considering that the energy consumption is closely related with communication cost, we can safely conclude that our protocols are more energy efficient than the existing work, which makes our protocols more practical for mobile applications.

4.6 Conclusion

While mobile social networking enables a large number of exciting applications, it also arouses serious privacy concerns. In this work, we introduce a privacy-preserving profile matching mechanism for secure friend discovery in mobile social networks. We allow the initiator to set his/her customized preference-profile and privacy protection level. By taking advantage of the Bloom filter, most unmatched users are excluded to significantly decrease the computational overhead. By adopting the Attribute Based Encryption, the profile privacy is preserved for the exactly matched users from other unmatched users as they cannot obtain much useful information. A secure verifiable communication channel is also established at the same time when the exactly matched users are found. We also consider the collusion attacks among partially matched users, and propose an enhanced mechanism to tackle such threats. We believe our design is effective, flexible, and complete in security protection and efficient in profile matching.

Chapter5

Cooperative Fine-grained Privacy Preserving Friend Discovery

In our work so far, we have developed two novel algorithms for privacy preserving friend discovery in MSNs, addressing both location privacy and profile privacy. Extensive theoretical analysis and experimental study have been conducted, and the results indicate that our schemes are feasible and effective for privacy-preserving friend discovery in mobile social networks.

These schemes can only enable coarse-grained private matching. In this chapter, we investigate and develop methods for fine-grained private profile matching. Fine grained personal profiles have advantages over traditional coarse-grained ones in the sense that fine-grained personal profiles enable finer differentiation among the users having different levels of interest in the same attribute. The accompanying challenge is how to ensure the privacy of fine-grained profile.

To tackle such a challenge, we propose a novel cooperative framework for the fine-grained privacy preserving friend discovery. Besides an introduction of the basic mechanism, we also propose two enhanced mechanisms by taking collusion attack and verifiability into consideration.

Next, we will give a detailed description of each mechanism.

5.1 Algorithm Design

We consider Alice with profile $u =< u_1, ..., u_d >$ and Bob with profile $v =< v_1, ..., v_d >$ as two exemplary users. To create a fine-grained personal profile, Alice/Bob selects an integer u_i/$v_i \in [0, \gamma - 1]$ to indicate her/his level of interest. As a fixed system parameter, γ could be a small integer, say 5 or 10, which should be sufficient to differentiate users' interest level. Let F denote a set of candidate matching metrics defined by the

application, where each $f \in F$ is a function over two personal profiles that measures their similarity.

Basically, the most straightforward matching metric is the ℓ_1 distance (also called the Manhattan distance), which is usually converted into the ℓ_2 distance between the unary representations of **u** and **v** [87], and then compute the ℓ_2 distance using a secure dot-product protocol.

In particular, for $\forall x \in [0, \gamma - 1]$, we define a binary vector $h(x) = (x_1, ..., x_{\gamma-1})$, where x_i is equal to one for $1 \leqslant i \leqslant x$ and zero for $x < i \leqslant \gamma - 1$. We also abuse the notation by defining binary vectors $\widehat{u} = h(u) = (h(u_1), ..., h(u_d)) = (\widehat{u}_1, ..., \widehat{u}_{(\gamma-1)d)})$ and $\widehat{v} = h(v) = (h(v_1), ..., h(v_d)) = (\widehat{v}_1, ..., \widehat{v}_{(\gamma-1)d)})$. It follows that

$$\begin{aligned} \ell_1(u, v) &= \sum_{i=1}^{d} |u_i - v_i| \\ &= \sum_{i=1}^{(\gamma-1)d} |\widehat{u}_i - \widehat{v}_i| \\ &= \ell_2^2(\widehat{u}, \widehat{v}) \end{aligned} \tag{5.1}$$

$$\begin{aligned} \ell_2^2(\widehat{u}, \widehat{v}) &= \sum_{i=1}^{(\gamma-1)d} |\widehat{u}_i - \widehat{v}_i|^2 \\ &= \sum_{i=1}^{(\gamma-1)d} \widehat{u}_i^2 - 2 \sum_{i=1}^{(\gamma-1)d} \widehat{u}_i \widehat{v}_i + \sum_{i=1}^{(\gamma-1)d} \widehat{v}_i^2 \\ &= \sum_{i=1}^{(\gamma-1)d} \widehat{u}_i^2 - 2\widehat{\mathbf{u}} \cdot \widehat{\mathbf{v}} + \sum_{i=1}^{(\gamma-1)d} \widehat{v}_i^2 \end{aligned}$$

Since Alice and Bob are aware of $\sum_{i=1}^{(\gamma-1)d} \widehat{u}_i^2$ and $\sum_{i=1}^{(\gamma-1)d} \widehat{v}_i^2$, respectively, we just need a secure dot-product to compute $\widehat{\mathbf{u}} \cdot \widehat{\mathbf{v}}$.

In addition, dot-product can be used to compute many other proximity measures. For

example, given a global list of attributes, each user is assigned a vector with 0s or 1s, where 0 means that the user does not have the attribute and 1 otherwise. The dot-product of two vectors essentially captures the number of matches between two users and can also be used to predict new friendship. As another example, consider two feature vectors **u** and **v**. If we normalize them to have a unit length, the dot-product $< u, v >$ gives the cosine similarity between these two feature vectors.

Based on these observations, we shall focus on the secure dot-product without loss of generality in this work.

5.1.1 Basic Mechanism

We first describe a simple and extremely efficient scheme for the dot-product function. It provides practical solutions to common problems like weighted average, hamming distance, etc.

We introduce the basic mechanism to compute dot-product between two vectors. We are able to compute the actual value of a dot-product, not just whether it is zero or not. We assume that the computation is conducted between the initiator Alice P_a and the candidate Bob P_b owning private vector u and v, respectively, with the help of another user P_c.

We first provide an overview of the construction. To encode an input vector u, the initiator Alice chooses a random vector r of the same dimension as u. She then outputs $x = u + r$ and sends x to Bob P_b, while the random noise r is sent to Helper P_c. Note that x and r form an additive secret sharing of u, and neither by itself contains any information about u. As a result, the private vector u is kept confidential to the initiator, as long as the candidate P_b and the helping user P_c do not collude with each other. The data share distribution of the basic scheme is illustrated in Figure 5.1.

After receiving x from the initiator, the candidate P_b locally computes $< v, x >$. Apparently, we have

$$< v, x > - < v, r > = < v, u > \tag{5.2}$$

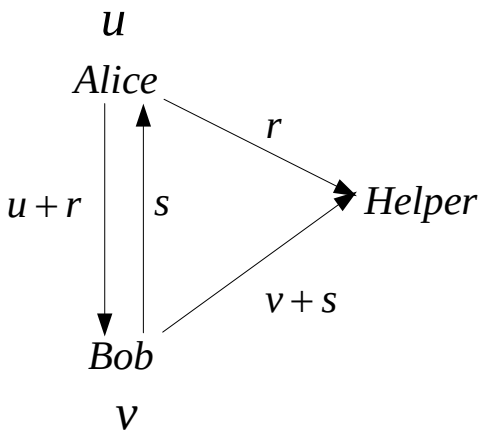


Figure 5.1: An illustration of data share distribution in the Basic Mechanism.

To obtain dot-product $< v, u >$, the candidate P_b needs to add the negative of the value of $< v, r >$, which is calculated with the help of P_c. Next, we introduce how to obtain $< v, r >$ at the user P_c.

P_c has already received r from the initiator in the previous step. It needs to get v to obtain $< v, r >$. However, v is private to P_b, which needs to be protected. Similar to the initiator, the candidate P_b chooses a noise s of the same dimension as v, and sends $y = v+s$ to the helping user P_c. By this way the privacy of v is also preserved. Meanwhile, the noise vector s is sent to the initiator P_a.

Next, P_c locally computes $< y, r >=< v + s, r >$, while P_a locally computes $< s, r >$ and returns it to P_c. Finally, P_c is able to obtain $< v, r >$ as follows:

$$< y, r > - < s, r >=< v + s, r > - < s, r >=< v, r > \tag{5.3}$$

After getting $< v, r >$, P_c sends it to the candidate P_b, and P_b obtains the dot-product $< v, u >$ according to (5.2).

Note that all the transmitted data are either noise or a vector mixed with noise; thus no private data is revealed. In other words, we carefully assign partial information to each

party by adopting the concept of information hiding to make sure that the profile privacy can not be inferred, yet the dot-product can be obtained.

Furthermore, this scheme can exploit the sparsity of a vector as the computation is proportional to the number of non-zero elements. In fact, even if all vector elements are non-zero, this construction is extremely fast, as shown by our performance evaluation.

We summarize our basic mechanism in Algorithm 5.1.

Algorithm 5.1: The Basic Scheme of Cooperative Secure dot-product

Alice has private vector u, Bob has private vector v.
1. Data share distribution:
1) Alice chooses a noise vector r, then sends the mixed vector $x = u + r$ to Bob, and sends r to the helper node.
2) Bob chooses a noise vector s, then sends the mixed vector $y = v + s$ to the Helper, and sends s to Alice.
2. Computation
1) Alice computes dot-product $< s, r >$ and sends it to Helper.
2) Bob computes dot-product $< v, u + r >$.
3) Helper computes dot-product $< v + s, r >$, then minus the received $< s, r >$ to get $< v, r >$, which is sent to Bob.
3. Reconstruction
Bob receives $< v, r >$ from Helper, and finally gets $< v, u >=< v, u+r > - < v, r >$.

5.1.2 Collusion Resistant Mechanism

In the basic mechanism, a helping user is involved in secure dot-product computation. Specifically, the private vector is concealed by adding a certain amount of noise. Then the noise vector and the resulted vector is separately distributed to two different users. Neither of them reveals private information of the original vector by itself. However, if the helping node colludes with P_a or P_b, the private vector can be obtained. A possible extension to improve the robustness is to slice the noise into several pieces, and send each piece to a helping node. In this way, many different users are involved, and a collusion attack will succeed if and only if all the involved users collude together, which is not easy when the number of helping users is large.

Such a countermeasure requires more helping users to prevent the collusion attack, say k different users $\{P_1, P_2, ..., P_k\}$. Next, we give a detailed description of the scheme involving multiple helping users. Again, we assume that the initiator P_a has a private vector u, and a candidate user P_b has a private vector v.

We first consider the case of the candidate Bob trying to collude with the helping users, and then introduce a mechanism that prevents such attacks.

The initiator: Instead of sending r to the helping user P_c, the initiator slices it into k pieces $\{r_1, r_2, ..., r_k\}$ and sent each to one of k different users $\{P_1, P_2, ..., P_k\}$. Note that $\sum_{i=1}^{k} r_i = r$.

The candidate: The candidate would send the mixed vector $y = v + s$ to each helping node P_i. At the same time, the noise vector s is sent to the initiator.

The helping nodes: Each helping node P_i receives a noise piece r_i from the initiator, and the mixed vector y from the candidate. Then P_i locally computes $< y, r_i >=< v + s, r_i >$. All these results are sent to the candidate.

The data share distribution of this process is illustrated in Figure 5.2.

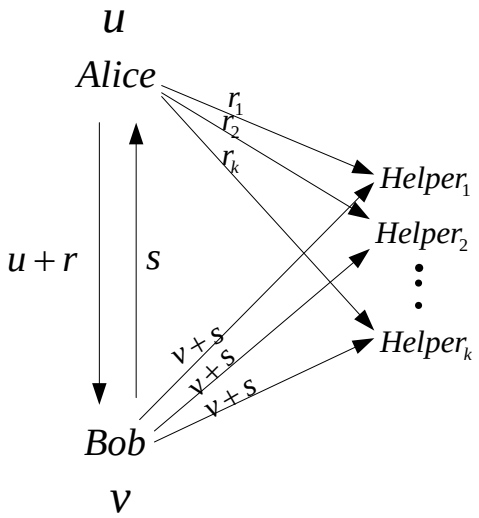


Figure 5.2: An illustration of data share distribution in the case of Bob colluding with helping users.

As the candidate P_b receives $u + r$ from P_a before, it computes $< v, u + r >$ locally. After receiving all the partial results from k different helping nodes, the candidate can obtain $\sum_{i=1}^{k} < v + s, r_i >=< v + s, r >$. Meanwhile the initiator computes dot-product $< s, r >$ and sends it back to the candidate. At last, the candidate obtains the dot-product $< v, r >=< v + s, r > - < s, r >$, then $< v, u >=< v, u + r > - < v, r >$.

Similarly, to prevent the initiator Alice to collude with the helping node, we slice Bob's noise s into k pieces with $\sum_{i=1}^{k} s_i = s$, send $v + s_i$ to each helping node P_i. So the partial results returned from the helping nodes will be $< v + s_i, r >$, which will result in the sum as $sum_{i=1}^{k} < v + s_i, r >= k < v, r > + < s, r >$. The initiator will conduct the same procedure as in the basic mechanism except that the noise r is sent to k different helping nodes. This process is illustrated in Figure 5.3.

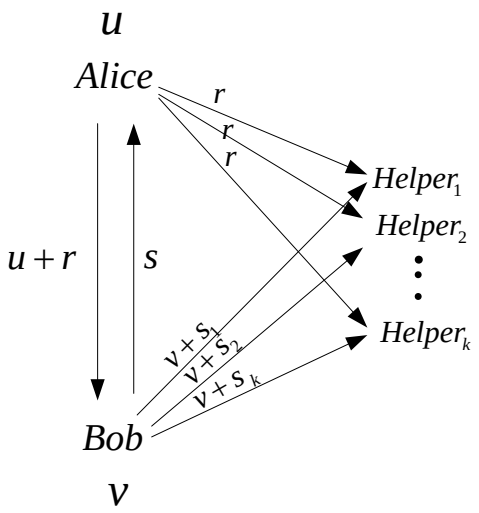


Figure 5.3: An illustration of data share distribution in the case of Alice colluding with helping users.

Eventually, the candidate will obtain i) $< v, u + r >$ computed locally; ii) $< s, r >$ returned from the initiator; and iii) $k < v, r > + < s, r >$ by summing up all the partial results from the helping nodes. Then the following calculations are performed:

(1) $\frac{k<v,r>+<s,r>)-<s,r>}{k} =< v, r >$

(2)$< v, u + r > - < v, r >=< v, u >$

Higher level of protection is achieved in the expense of larger communication overhead. But the computation involves only basic additions and multiplications. No expensive homomorphic encryption or exponentiation is used. Please note that we consider the case when the helping node colludes with either initiator or candidate. The case of the helping node colluding with both the initiator and the candidate simultaneously is unusual, hence is not considered in this work.

People may argue that the collusion attack is not completely tackled in the specific case when all nodes collude together, but this problem commonly exists in all schemes where cooperation among different users is necessary. Moreover, a functional system should have more than 50% of benign users. Another possible extension is to assume that the helping user P_c is a trusted authority, which will legitimately execute the whole procedure as required and no collusion will be conducted.

5.1.3 Verifiable Mechanism

In the above schemes, the initiator sends an invitation to the candidate. After all the message exchanges, the candidate obtains the secure dot-product of the initiator's vector and its private vector. Apparently, the final result of dot-product is only known by one party. Next, we introduce a protocol whereby the result is verifiable by both parties. For both party to obtain the dot-product, the initiator and the candidate run two separate instances of the protocol in parallel.

To realize verification, some slight modification based on the basic mechanism is required. The detailed algorithm is described as follows:

Step 1:

The initiator Alice generates a noise vector r and mixes it with its private vector u, then sends the resulted $(u + r)$ to candidate Bob. Similarly, Bob mixes its vector v with noise vector s and sends $(v + s)$ to Alice. After receiving the corresponding mixed vector, Alice and Bob compute $< u, v + s >$ and $< v, u + r >$, respectively.

Alice: $< u, v + s >$ (a)

Bob: $< v, u + r >$ (a')

Step 2:

Alice chooses another noise vector α, then sends both r and the new mixed vector $u+\alpha$ to a helping node P_{h1}. Similarly, Bob chooses a new noise vector β, then sends $(s, v+\beta)$ to the same helping node P_{h1}. Thus the helper P_{h1} can calculate $< u + \alpha, s >$ and $< v + \beta, r >$.

P_{h1}: $< u + \alpha, s >$ (b)

$< v + \beta, r >$ (b')

Step 3:

Alice further sends two noise vectors (r, α) to a second helping node P_{h2}. Bob does the same by sending (s, β) to P_{h2}. Then P_{h2} can have the dot-product $< \alpha, s >$ and $< \beta, r >$

P_{h2}: $< \alpha, s >$ (c)

$< \beta, r >$ (c')

Two helpers P_{h1} and P_{h2} send (b) and (c) to Alice, who can finally get dot-product $< u, v >= (a) - ((b) - (c))$. The same procedure is conducted for Bob to get the dot-product $< v, u >= (a') - ((b') - (c'))$.

The data share distribution of the verifiable scheme is illustrated in Figure 5.4.

Now both Alice and Bob obtain the dot-product. A naive verification approach might be to encrypt the dot-product and send it to each other, then the other party decrypts it and compares with the locally computed result for consistency.

5.2 Performance Evaluation

5.2.1 Security Analysis

In our scheme, a private vector contains the private attribute that needs to be protected. To achieve privacy preservation, the original private vectors are never sent out, yet we are

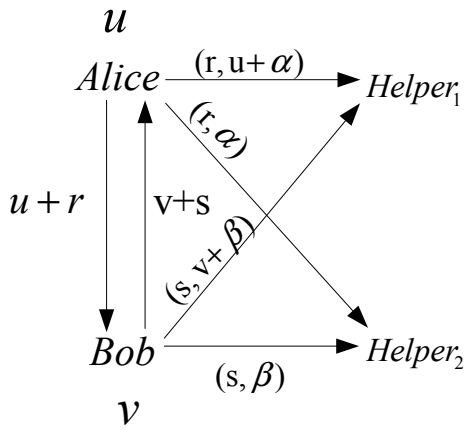


Figure 5.4: An illustration of the data share distribution in the Verifiable Mechanism.

Table 5.2: Computation cost comparison

Protocol	Offline Comp	Online Comp
Zhang *et al.* [87]	$4\gamma d + 4\ exp_1$, $2\gamma d + 2\ mul_2$	$8exp_2$, $8exp_1$, $2d + 6\ mul_2$
Dong *et al.* [11]	$4d\ exp_1$, $2d\ mul_2$	$4\ exp_2$, $8d + 8\ exp_1$, $4d + 6\ mul_2$
Basic Mechanism	0	$3d\ mul_1$
Collusion Resistant Mechanism	0	$kd + 2d\ mul_1$
Verifiable Mechanism	0	$6d\ mul_1$

Table 5.3: Execution time of different operations

mul_1	mul_2	exp_1	exp_2
8×10^{-5} ms	2.4×10^{-4} ms	40 ms	250 ms

able to calculate the dot-product between two private vectors for similarity measurement.

While most existing work resorts to the expensive cryptographic algorithms to protect the privacy, we hide the privacy by adding random noise to it. Only the noise and the vector mixed with the noise are sent out instead of the original vector. In this way, the privacy is effectively protected. And the computation of the private vectors' dot-product is based on the partial results computed by different parties, while the partial results reveal no private information. In summary, the privacy is preserved because the information sent out reveals no private information, also the computed results of each party reveals no private information.

We also consider collusion attack and verifiability and propose enhanced mechanisms for each case. In all these mechanisms, we guarantee that neither the candidate nor the helping nodes can infer any private information.

5.2.2 Performance Analysis

In this section, we evaluate the computational overhead as well as the overall execution time of our protocols in contrast to the previous work. We use 1024 bits for each value of the vector.

Table 5.2 summarizes the theoretical performance of our protocols and the existing work proposed in [87], [11], which also falls in the category of secure dot-product. In this table, mul_1, mul_2, exp_1, and exp_2 denote a 1024-bit multiplication, a 2048-bit multiplication, a 1024-bit exponentiation, and a 2048-bit exponentiation, respectively; d is the number of attributes in a profile vector, which is set to be 15; k is the number of helping nodes in the verifiable mechanism, which is set to be 5; and γ is a small integer used to indicate the level of interest in the fine-grained profile matching, which is set to 5 as in [87]. We will study the impact of these parameters on the computational cost in this section.

Note that a complete matching process involves independent executions of private matching on different parties, e.g., the initiator and the candidate. Our protocols involve

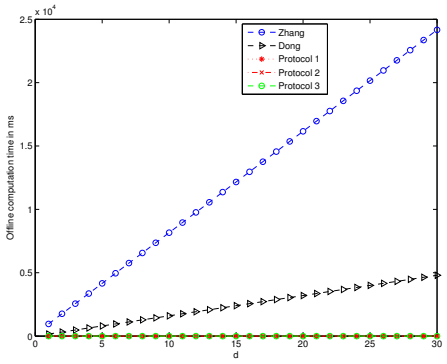
helping users as well. Thus we summarize the computational cost in Table 5.2 as the total cost of different parties, and we consider both offline and online computational cost.

Since our protocols do not involve expensive encryption algorithms, one can see that the offline computation cost is 0, while the two existing work involves exponentiations and multiplications for encryption before sending out the profile data. In addition, for online computation, our work is basically executing dot-product for several times, while the existing work is more about homomorphic encryption/decryption, and ciphertext computation. Thus the computational cost of dot-product in our protocols is based on mul_1; and the two existing work involves more expensive exp_2, exp_1 and mul_2.

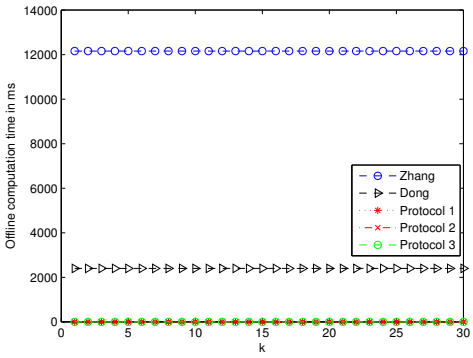
To further quantify the computational cost in terms of execution time, we follow the evaluation methodology in [87]. And we assume that each user is using a mobile device with a 400 MHz CPU. According to the benchmark test results in [6], it takes about 8×10^{-5} ms, 2.4×10^{-4} ms, 40 ms, and 250 ms to perform one mul_1, one mul_2, one exp_1, and one exp_2, respectively, as shown in Table 5.3. We also assume that two mobile devices communicate with each other through 802.11b interface with a transmission rate 2 Mbit/s.

Figure 5.5 shows comparison of the offline computation cost. Our three protocols are denoted as Protocol 1, 2, 3 in the figure. In Figure 5.5(a), we study the case when d varies, while other parameters are set to be typical values as mentioned before: $k = 5$ and $\gamma = 5$. As explained before, our protocols have no offline computation cost. And it is not surprising to see that the offline computation cost of two methods in existing work is proportional to d. Also, we can see that both methods in existing work incur much more offline computation cost. Figure 5.5(b) and 5.5(c) study the case when k and γ vary while $d = 15$. We can see that these two figures demonstrate the similar results that existing work is more expensive in term of offline commutation cost, while our work has no offline cost.

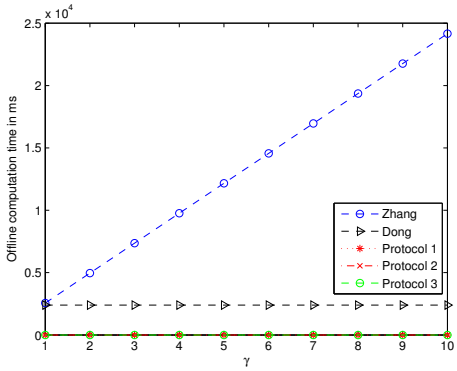
Figure 5.6 shows the comparison results of the online computational cost. It is clear that all of our protocols incur significantly lower overhead than the existing work. The main



(a) Change d

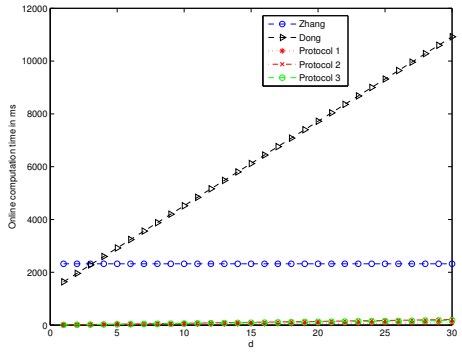


(b) Change k

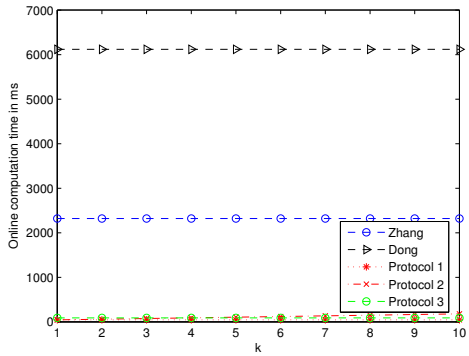


(c) Change γ

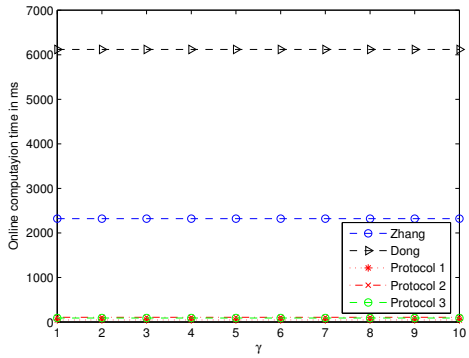
Figure 5.5: Offline computation cost.



(a) Change d



(b) Change k



(c) Change γ

Figure 5.6: Online computation cost.

reason lies in that the expensive operations exp_2 (2048-bit exponentiation) and mul_2 (2048-bit multiplication) dominate the online computational cost of the existing work, while our work only involves simple 1024-bit multiplication operations.

It is worth noting that since all our protocols involve standard multiplications only, they are readily available for implementation on smartphones, tablets, and other modern mobile devices. Such engineering efforts are part of our ongoing work.

5.3 Conclusion

With increasing popularity of mobile social networks, a large number of exciting applications are enabled. However, serious privacy concerns have hindered the wide adoption of these applications. It is important to develop secure and efficient protocols to enable profile matching for friend discovery. In this work, we introduce a novel framework to compute the similarity of two profiles as dot-product. Instead of resorting to expensive cryptographic algorithms, we propose to preserve the data privacy by adding a certain amount of noise to it, and carefully assign different shares of the privacy to different parties. While each sharing reveals no private information, computation based on them will finally lead to the desired dot-product of two private vectors. We first give a detailed introduction of the basic mechanism, then propose two enhanced mechanisms by considering collusion resistance and verifiability. We believe that our design is effective in security protection and efficient in computational cost.

Chapter6

Conclusions

This dissertation research addresses the key issue of privacy-preserving friend discovery in mobile social networks. Different privacy-preserving mechanisms have been proposed according to different design principles.

The first work proposes to do private profile matching based on common encounters. We take advantage of the fact that sharing encounters indicate common activities and interests. The friend discovery is realized while the location privacy and encounter privacy are guaranteed. Since a symmetric cryptographic key is established in an ad-hoc model between two users whenever they encounter with each other, no trusted centralized server is involved. We believe our design is more suitable to the dynamic property of mobile social networks. The performance evaluation and theoretical analysis have justified the effectiveness and feasibility of our scheme.

The second work proposes to adopt Bloom filter for private profile matching. By transferring the attribute profile into random 0/1 stream, Bloom filter helps to preserve privacy. In the meantime, Bloom filter is used as an effective tool to exclude most unmatched users. This significantly decreases the computational overhead. By further adopting the Attribute Based Encryption, the profile privacy is preserved to the users with all the required attributes. In addition, we allow the initiator to set his/her customized preference-profile and privacy protection level to make the scheme more flexible. We also consider the verifiability and collusion attacks and propose enhanced mechanism to make the scheme more secure. Generally speaking, our design is effective, flexible, and complete in security protection and efficient in profile matching.

The third work proposes to do the fine-grained private profile matching in a highly efficient way. Without involving any expensive cryptographic algorithms, we add a certain

amount of noise to the private data. And by leveraging the concept of information hiding, different shares of the privacy is carefully assigned to different parties, while the information of each party is strictly limited. In this way, the matching result can be obtained in a cooperative manner with each sharing revealing no private information. We first give a detailed introduction of the basic mechanism. To make the scheme more complete, we further propose two enhanced mechanisms by considering collusion resistance and verifiability. The evaluation results show that our design is effective in security protection and efficient in computational cost.

In summary, we study the problem of privacy preserving friend discovery in mobile social networks from different angles in this dissertation. And three novel solutions are proposed for three different scenarios under the same topic. Security analysis and performance evaluation have been conducted, and the results indicate that our protocols are effective and efficient.

Chapter7

Future Work

Our research so far mainly focuses on the procedure of friend discovery to establish a social relationship between two users. For our future work, we will work on the privacy protection issues after a relationship is established.

More and more people are utilizing social network services, and publishing their personal attributes about themselves (e.g., age, gender, and interests) on line. Unlike traditional personal home pages, these networks also allow users to publish their relationships with friends. Given the huge amount of personal data and social relationships available, it is possible to infer unpublished private information by using learning algorithms on released data. Such inference attacks [2, 18, 81] become increasingly challenging as social network services are more popular and more personal information is released.

In an inference attack, a user's private information can be intelligently inferred by combining pieces of seemingly innocuous or unrelated information. Based on the observation that a user's private attribute can be reflected from his released attributes or his/her friendship links, many researchers consider ways to infer private information by applying Bayesian inference [26,33,34,67]. And the defense methods for inference attacks mainly consider sanitizing attributes from a user's on-line profile or its link details with friends [27, 49, 65].

However, as long as the user is connected in a network, its privacy is not safe because the user has no control over his/her connected users. And these users will put a risk on privacy leakage. Therefore, removing attributes and links seems an obvious way to alleviate inference attacks, but it is not 100% effective, unless this user is totally isolated with no connections, which is not an option for a normal user.

Besides, even removing a small faction of attributes and links would decrease the

usability of online social networks. Because people's profile and friendship information is not complete, the quality of social network service would be weakened.

How to balance the tradeoff between usability and privacy or preserve privacy without weakening the service is a very challenging problem. We plan to design a cooperative mechanism to effectively protect each user's privacy while making sure the usability of the social service is not weakened. The main observation of our mechanism is that the protection of inference attack should be considered from a global perspective.

Basically, the users will need to reach a certain agreement for the sake of privacy protection. In the mean time, they may have different perspectives and requirements for the privacy.

- We assume the users have reached a certain agreement on the cooperation mechanism. To be more specific, a user will protect the other users' privacy, in return his/her own privacy will be protected by the other users. If nobody reveals private information, they will all be safe. If anybody breaks the rule, the breaker itself will not be safe. Providing incentives for the users to cooperate is an important topic. There are extensive mechanisms [3, 15, 84] addressing the cooperation incentives [14, 70, 71].

- The users' difference should be considered. People may have different perspectives for protecting privacy. For example, some people may be more concerned about certain attributes, while other people do not care. Or the information people care about could be very different. In such cases, even if a user pays extra attention to his/her privacy, his/her friends may still give out some clues, especially when information is inferred across heterogenous networks. As a simple example, a user wants to keep his birthday secret. But his friends may post "Happy Birthday" comments on that day.

In summary, we believe a cooperative defense method is needed for protecting a user's

privacy from a community's perspective. As all the nodes in a social network are connected with each other, we believe it would be better to take the network as a whole community. And the users within the community should cooperate with each other to protect certain information, while their differences are taken into consideration.

Bibliography

[1] Rudolf Ahlswede and Imre Csiszar. Common randomness in information theory and cryptography. i. secret sharing. *Information Theory, IEEE Transactions on*, 39(4):1121–1132, 1993.

[2] Seyed Hossein Ahmadinejad, Mohd Anwar, and Philip Fong. Inference attacks by third-party extensions to social network systems. 2010.

[3] Scott Barrett. Why cooperate?: the incentive to supply global public goods. *OUP Catalogue*, 2010.

[4] Aaron Beach, Mike Gartrell, Sirisha Akkala, Jack Elston, John Kelley, Keisuke Nishimoto, Baishakhi Ray, Sergei Razgulin, Karthik Sundaresan, Bonnie Surendar, et al. Whozthat? evolving an ecosystem for context-aware mobile social networks. *Network, IEEE*, 22(4):50–55, 2008.

[5] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 321–334. IEEE, 2007.

[6] Siddharth Bhatt, Radu Sion, and Bogdan Carbunar. A personal mobile drm manager for smartphones. *Computers & Security*, 28(6):327–340, 2009.

[7] Garrett Brown, Travis Howe, Micheal Ihbe, Atul Prakash, and Kevin Borders. Social networks and context-aware spam. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 403–412. ACM, 2008.

[8] Richard Chow and Philippe Golle. Faking contextual data for fun, profit, and privacy. In *Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pages 105–108. ACM, 2009.

[9] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography and Data Security*, pages 143–159. Springer, 2010.

[10] Paresh Dhakan and Ronaldo Menezes. The role of social structures in mobile ad-hoc networks. In *Proceedings of the 43rd annual Southeast regional conference-Volume 2*, pages 59–64. ACM, 2005.

[11] Wei Dong, Vacha Dave, Lili Qiu, and Yin Zhang. Secure friend discovery in mobile social networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 1647–1655. IEEE, 2011.

[12] Nathan Eagle and Alex Pentland. Social serendipity: Mobilizing social software. *Pervasive Computing, IEEE*, 4(2):28–34, 2005.

[13] Donald E Eastlake, Stephen D Crocker, and Jeffrey I Schiller. Randomness requirements for security. Technical report, RFC 1750, Dec, 1994.

[14] Harri Ehtamo and Raimo P Hämäläinen. A cooperative incentive equilibrium for a resource management problem. *Journal of Economic Dynamics and Control*, 17(4):659–678, 1993.

[15] Ernst Fehr and Urs Fischbacher. Why social preferences matter–the impact of non-selfish motives on competition, cooperation and incentives. *The economic journal*, 112(478):C1–C33, 2002.

[16] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 427–438. IEEE, 1987.

[17] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology-EUROCRYPT 2004*, pages 1–19. Springer, 2004.

[18] Arik Friedman and Assaf Schuster. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–502. ACM, 2010.

[19] Bugra Gedik and Ling Liu. Location privacy in mobile systems: A personalized anonymization model. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 620–629. IEEE, 2005.

[20] Bugra Gedik and Ling Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *Mobile Computing, IEEE Transactions on*, 7(1):1–18, 2008.

[21] Craig Gentry et al. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.

[22] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 1998.

[23] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. ACM, 2006.

[24] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.

[25] Tzipora Halevi and Nitesh Saxena. On pairing constrained wireless devices based on secrecy of auxiliary channels: The case of acoustic eavesdropping. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 97–108. ACM, 2010.

[26] Jianming He, Wesley W Chu, and Zhenyu Victor Liu. Inferring privacy information from social networks. In *Intelligence and Security Informatics*, pages 154–165. Springer, 2006.

[27] Raymond Heatherly, Murat Kantarcioglu, and Bhavani Thuraisingham. Preventing private information inference attacks on social networks. *Knowledge and Data Engineering, IEEE Transactions on*, 25(8):1849–1862, 2013.

[28] Monika R Henzinger, Rajeev Motwani, and Craig Silverstein. Challenges in web search engines. In *ACM SIGIR Forum*, volume 36, pages 11–22. ACM, 2002.

[29] Mark Hillery, Vladimír Bužek, and André Berthiaume. Quantum secret sharing. *Physical Review A*, 59(3):1829, 1999.

[30] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. Preserving privacy in gps traces via uncertainty-aware path cloaking. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 161–171. ACM, 2007.

[31] Xiaonan Hu, Terry Chu, Victor Leung, Edith Ngai, Philippe Kruchten, and Henry Chan. A survey on mobile social networks: Applications, platforms, system architectures, and future research directions. 2015.

[32] Chung-Ming Huang, Kun-chan Lan, and Chang-Zhou Tsai. A survey of opportunistic networks. In *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on*, pages 1672–1677. IEEE, 2008.

[33] John P. Huelsenbeck, Fredrik Ronquist, et al. Mrbayes: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755, 2001.

[34] John P Huelsenbeck, Fredrik Ronquist, Rasmus Nielsen, and Jonathan P Bollback. Bayesian inference of phylogeny and its impact on evolutionary biology. *science*, 294(5550):2310–2314, 2001.

[35] Lee Humphreys. Mobile social networks and social practice: A case study of dodgeball. *Journal of Computer-Mediated Communication*, 13(1):341–360, 2007.

[36] Ioannis Ioannidis, Ananth Grama, and Mikhail Atallah. A secure protocol for computing dot-products in clustered and distributed environments. In *Parallel Processing, 2002. Proceedings. International Conference on*, pages 379–384. IEEE, 2002.

[37] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, pages 153–181. Springer, 1996.

[38] Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. Preventing location-based identity inference in anonymous spatial queries. *Knowledge and Data Engineering, IEEE Transactions on*, 19(12):1719–1733, 2007.

[39] Bishal Raj Karki, Arto Hämäläinen, and Jari Porras. Social networking on mobile environment. In *Proceedings of the ACM/IFIP/USENIX Middleware'08 Conference Companion*, pages 93–94. ACM, 2008.

[40] Nipendra Kayastha, Dusit Niyato, Ping Wang, and Ekram Hossain. Applications, architectures, and protocol design issues for mobile social networks: A survey. *Proceedings of the IEEE*, 99(12):2130–2158, 2011.

[41] Adam Kirsch and Michael Mitzenmacher. Less hashing, same performance: Building a better bloom filter. In *Algorithms–ESA 2006*, pages 456–467. Springer, 2006.

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

[43] Balachander Krishnamurthy and Craig E Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 7–12. ACM, 2009.

[44] Abhishek Kumar, Jun Xu, and Jia Wang. Space-code bloom filter for efficient per-flow traffic measurement. *Selected Areas in Communications, IEEE Journal on*, 24(12):2327–2339, 2006.

[45] Kin-Wah Kwong, Augustin Chaintreau, and Roch Guerin. Quantifying content consistency improvements through opportunistic contacts. In *Proceedings of the 4th ACM workshop on Challenged networks*, pages 43–50. ACM, 2009.

[46] Ming Li, Ning Cao, Shucheng Yu, and Wenjing Lou. Findu: Privacy-preserving personal profile matching in mobile social networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 2435–2443. IEEE, 2011.

[47] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *Parallel and Distributed Systems, IEEE Transactions on*, 24(1):131–143, 2013.

[48] Zi Lin, Denis Foo Kune, and Nicholas Hopper. Efficient private proximity testing with gsm location sketches. In *Financial Cryptography and Data Security*, pages 73–88. Springer, 2012.

[49] Jack Lindamood, Raymond Heatherly, Murat Kantarcioglu, and Bhavani Thuraisingham. Inferring private information using social network data. In *Proceedings of the 18th international conference on World wide web*, pages 1145–1146. ACM, 2009.

[50] Rongxing Lu, Xiaodong Lin, Xiaohui Liang, and Xuemin Shen. A secure handshake scheme with symptoms-matching for mhealthcare social network. *Mobile Networks and Applications*, 16(6):683–694, 2011.

[51] Ben Lynn. *On the implementation of pairing-based cryptosystems*. PhD thesis, Stanford University, 2007.

[52] Justin Manweiler, Ryan Scudellari, and Landon P Cox. Smile: Encounter-based trust for mobile social services. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 246–255. ACM, 2009.

[53] Suhas Mathur, Wade Trappe, Narayan Mandayam, Chunxuan Ye, and Alex Reznik. Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 128–139. ACM, 2008.

[54] Ueli M Maurer. A universal statistical test for random bit generators. *Journal of cryptology*, 5(2):89–105, 1992.

[55] Ueli M Maurer. Secret key agreement by public discussion from common information. *Information Theory, IEEE Transactions on*, 39(3):733–742, 1993.

[56] Emiliano Miluzzo, Nicholas D Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B Eisenman, Xiao Zheng, and Andrew T Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 337–350. ACM, 2008.

[57] Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, and Dan Boneh. Location privacy via private proximity testing. In *NDSS*, 2011.

[58] Fawad Nazir, Jiaxin Ma, and Aruna Seneviratne. Time critical content delivery using predictable patterns in mobile social networks. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 1066–1073. IEEE, 2009.

[59] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 195–203. ACM, 2007.

[60] Di Qiu, Dan Boneh, Sherman Lo, and Per Enge. Robust location tag generation from noisy location data for security applications. In *The Institute of Navigation International Technical Meeting*. Citeseer, 2009.

[61] Di Qiu, Sherman Lo, Per Enge, Dan Boneh, and Ben Peterson. Geoencryption using loran. In *The Institute of Navigation International Technical Meeting*, 2007.

[62] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73–85. ACM, 1989.

[63] Theodore Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2001.

[64] Ramprasad Ravichandran, Michael Benisch, Patrick Gage Kelley, and Norman M Sadeh. Capturing social networking privacy preferences. In *Privacy Enhancing Technologies*, pages 1–18. Springer, 2009.

[65] Murat Kantarcioglu Raymond Heatherly and Bhavani Thuraisingham. Preventing private information inference attacks on social networks. Technical report, Technical Report UTDCS-03-09, University of Texas at Dallas, 2009.

[66] Patrick F Riley. The tolls of privacy: An underestimated roadblock for electronic toll collection usage. *Computer Law & Security Review*, 24(6):521–528, 2008.

[67] Fredrik Ronquist and John P Huelsenbeck. Mrbayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19(12):1572–1574, 2003.

[68] Norman Sadeh, Jason Hong, Lorrie Cranor, Ian Fette, Patrick Kelley, Madhu Prabaker, and Jinghai Rao. Understanding and capturing people's privacy policies in a mobile social networking application. *Personal and Ubiquitous Computing*, 13(6):401–412, 2009.

[69] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473. Springer, 2005.

[70] Robert E Slavin. When does cooperative learning increase student achievement? *Psychological bulletin*, 94(3):429, 1983.

[71] Robert E Slavin. When and why does cooperative learning increase achievement. *The RoutledgeFalmer reader in psychology of education*, 1:271–293, 2004.

[72] Haoyu Song, Sarang Dharmapurikar, Jonathan Turner, and John Lockwood. Fast hash table lookup using extended bloom filter: an aid to network processing. *ACM SIGCOMM Computer Communication Review*, 35(4):181–192, 2005.

[73] Richard Sproat, Alan W Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333, 2001.

[74] Nikolaos Vastardis and Kun Yang. Mobile social networks: Architectures, social properties, and key research challenges. *Communications Surveys & Tutorials, IEEE*, 15(3):1355–1371, 2013.

[75] Marco Von Arb, Matthias Bader, Michael Kuhn, and Roger Wattenhofer. Veneta: Serverless friend-of-friend detection in mobile social networking. In *Networking and Communications, 2008. WIMOB'08. IEEE International Conference on Wireless and Mobile Computing,*, pages 184–189. IEEE, 2008.

[76] Shengling Wang, Min Liu, Xiuzhen Cheng, Zhongcheng Li, Jianhui Huang, and Biao Chen. Opportunistic routing in intermittently connected mobile p2p networks. *Selected Areas in Communications, IEEE Journal on*, 31(9):369–378, 2013.

[77] Shengling Wang, Min Liu, Xiuzhen Cheng, and Min Song. Routing in pocket switched networks. *Wireless Communications, IEEE*, 19(1):67–73, 2012.

[78] Shengling Wang, Xia Wang, Jianhui Huang, Rongfang Bie, and Xiuzhen Cheng. Analyzing the potential of mobile opportunistic networks for big data applications. In *Network Magzine*, page to appear. IEEE, 2014.

[79] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography–PKC 2011*, pages 53–70. Springer, 2011.

[80] Toby Xu and Ying Cai. Feeling-based location privacy protection for location-based services. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 348–357. ACM, 2009.

[81] Wanhong Xu, Xi Zhou, and Lei Li. Inferring privacy information via social relations. In *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*, pages 525–530. IEEE, 2008.

[82] Andrew C Yao. Protocols for secure computations. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE, 1982.

[83] Qingsong Ye, Huaxiong Wang, and Josef Pieprzyk. Distributed private matching and set operations. In *Information Security Practice and Experience*, pages 347–360. Springer, 2008.

[84] Chi Zhang, Xiaoyan Zhu, Yang Song, and Yuguang Fang. C4: A new paradigm for providing incentives in multi-hop wireless networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 918–926. IEEE, 2011.

[85] Lan Zhang, Xiang-Yang Li, and Yunhao Liu. Message in a sealed bottle: Privacy preserving friending in social networks. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, pages 327–336. IEEE, 2013.

[86] Rui Zhang, Jinxue Zhang, Yanchao Zhang, Jinyuan Sun, and Guanhua Yan. Privacy-preserving profile matching for proximity-based mobile social networking. *Selected Areas in Communications, IEEE Journal on*, 31(9):656–668, 2013.

[87] Rui Zhang, Yanchao Zhang, Jinyuan Sun, and Guanhua Yan. Fine-grained private matching for proximity-based mobile social networking. In *INFOCOM, 2012 Proceedings IEEE*, pages 1969–1977. IEEE, 2012.

[88] Haobin Zhong, Lingyan Bi, Zewei Feng, and Ning Li. Research on the design method of mobile social network services. In *Information Management, Innovation Management and Industrial Engineering, 2008. ICIII'08. International Conference on*, volume 2, pages 458–461. IEEE, 2008.