Message in a Sealed Bottle: Privacy Preserving Friending in Mobile Social Networks

Lan Zhang, *Member, IEEE,* Xiang-Yang Li, *Senior Member, IEEE,* Kebin Liu, *Member, IEEE,* Taeho Jung, *Student Member, IEEE,* and Yunhao Liu, *Senior Member, IEEE*

Abstract—Many proximity-based mobile social networks are developed to facilitate connections between any two people, or to help a user to find people with a matched profile within a certain distance. A challenging task in these applications is to protect the privacy of the participants' profiles and communications.

In this paper, we design novel mechanisms, when given a preference-profile submitted by a user, that search persons with matching-profile in decentralized mobile social networks. Meanwhile, our mechanisms establish a secure communication channel between the initiator and matching users at the time when a matching user is found. These techniques can also be applied to conduct privacy preserving keywords based search without any secure communication channel. Our analysis shows that our mechanism is privacy-preserving (no participants' profile and the submitted preference-profile are exposed), verifiable (both the initiator and any unmatched user cannot cheat each other to pretend to be matched), and efficient in both communication and computation. Extensive evaluations using real social network data, and actual system implementation on smart phones show that our mechanisms are significantly more efficient than existing solutions.

Index Terms—Privacy Preserving Profile Matching, Secure Communication, Decentralized Mobile Social Networks.

1 Introduction

A Boom in mobile hand-held devices greatly enriches the social networking applications. Many social networking services are available on mobile phones (*e.g.*, WeChat, QQ, Whatsup, JuiceCaster, MocoSpace and WiFace [31]) and majority of them are location-aware (*e.g.*, FourSquare, BrightKite, Loopt, Gypsii, meetMoi and frenzo). However, most of them are designed for facilitating people connections based on their real life social relationship [20], [26]. There is an increasing difficulty of befriending new people or communicating with strangers while protecting the privacy of real personal information [29].

Friending and communication are two important basic functions of social networks. When people join social networks, they usually begin by creating a profile, then interact with other users. The content of profile could be very broad, such as personal background, hobbies, contacts, places they have been to, etc. Profile matching is a common and helpful way to make new friends with mutual interests or experiences, find lost connections or search for experts [30]. Some applications help a user automatically find users with similar profile within a certain distance. For example, in the social network *Color*, people in close proximity (within 50 meters) can share photos automatically based on their similarity. MagnetU [1] matches one with nearby people for dating and friend-making. Small-talks [28] connects proximate users based on common interests. These applications use profiles to facilitate friending between proximate strangers and enable privacy preserving people searching to some extent.

Observe that in practice the mobile Internet connection may not always be available and it may incur high expense. Thus, in this work we focus on proximity-based decentralized mobile social networks (MSN) based on short-range wireless technologies such as WiFi and Bluetooth, *e.g.* [23], [31]. However the increasing privacy concern becomes a barrier for adopting MSN. People are unwilling to disclose personal profiles to arbitrary persons in physical proximity before deciding to interact with them. The insecure wireless communication channel and potentially untrusted service provider increase the risk of revealing private information.

Friending based on *private profile matching* allows two users to match their personal profiles without disclosing them to each other. There are two mainstreams of approaches to solve this problem. The first category treats a user's profile as a set of attributes and provides private attributes matching based on *private set intersection* (PSI) and *private cardinality of set intersection* (PCSI) [16], [27]. The second category considers a user's profile as a vector and measures the social proximity by *private vector dot product* [6], [10], [35]. They rely on public-key cryptosystem and homomorphic encryption, which results in expensive computation cost and usually requires a trusted third party. Multiple rounds of interactions are required to perform the presetting (*e.g.* exchange public keys) and private matching between each pair of users, which causes high communication cost. Moreover, most of those protocols are *unverifiable*: in the final step of these protocols, only one party learns the matching result and there lack efficient methods to verify the result. Furthermore, in these approaches, matched users and unmatched users all get involved in the expensive computation and learn their matching results (*e.g.* profile intersection) with the initiator. These limitations hinder the adoption of the SMC-related private matching methods in MSN.

A secure communication channel is equally important but often ignored in MSN. Although the matching process is

private, the following chatting may still be disclosed to the adversary and more privacy may be leaked. One simple solution is to build a secure communication channel using public key cryptosystem. This involves a trusted third party and key management, which is difficult to manage in decentralized MSN.

Facing these challenges, we first formally define the privacy preserving verifiable profile matching problem in decentralized MSN (Section 2). We then propose several protocols (Section 3) to address the privacy preserving profile matching and secure communication channel establishment in decentralized social networks without any presetting or trusted third party. We take advantage of the common attributes between matching users, and use it to encrypt a message with a secret channel key in it. In our mechanisms, only a matching user can decrypt the message. A privacy-preserving profile matching and secure channel construction are completed simultaneously with only one round of communication. The secure channel construction resists the Man-in-the-Middle attack by any unmatched users. Both precise and fuzzy matching/search in a flexible form are supported. The initiator can define a similarity threshold, the participant whose similarity is below the threshold learns nothing. A sequence of well-designed schemes make our protocols practical, flexible and lightweight, *e.g.*, a remainder vector is designed to significantly reduce the computation and communication overhead of unmatched users. Our profile matching mechanisms are also *verifiable* which thwart cheating about matching result. We also design a mechanism for location privacy preserved vicinity search based on our basic scheme. Compared to most existing works (Section 6) relying on the asymmetric cryptosystem and trusted third party, our protocols require no presetting and much less computation. Our methods can also be applied to conduct efficient privacy preserving keywords based search without any secure communication channel, *e.g.*, private image search and sharing [33], [32]. To the best of our knowledge, these are the first privacy-preserving verifiable profile matching protocols based on symmetric cryptosystem.

We analyze the security and performance of our mechanisms (Section 4). We then conduct extensive evaluations on the performances of our mechanisms using large scale social network data. Our results (Section 5) show that our mechanisms outperform existing solutions significantly. We also implement our protocols on laptop and mobile phone and measure the computation and communication cost in real systems. In our mobile-phone implementation, a user only needs about 1.3ms to generate a friending request. On average, it only takes a non-candidate user about 0.63ms and a candidate user 7ms to process this request.

2 System Model and Problem Definition

2.1 System Model

A user in a mobile ad hoc social networking system usually has a profile (a set of attributes). The attribute can be anything generated by the system or input by the user, including his/her location, places he/she has been to, his/her social groups, experiences, interests, contacts, keywords of his/her blogs, etc. According to our analysis of two well-known social networking systems (Facebook and Tencent Weibo [2]), more than 90% users have unique profiles. Thus for most users, the complete profile can be his/her fingerprint in social networks. The profile could be very useful for searching and friending people. But it is also very risky to reveal the fingerprint to strangers. Then, in most social networks, friending usually takes two typical steps: profile matching and communication. These applications cause a number of privacy concerns.

1) **Profile Privacy**: The profiles of all participants, including the initiator, intermediate relay users and the matched targets, should not be exposed without their consents. For participants, protecting their profiles is necessary and can reduce the barrier to participate in MSN. Note that, the exact location information is also a part of the user's profile privacy.
2) **Communication Security**: The messages between a pair of users should be transmitted through a secure communication channel. We emphasize that the secure communication channel establishment has been ignored in most previous works which address the private profile matching in decentralized MSN. In practice, after profile matching, more privacy, even profile information, may be exposed via communication through an insecure channel.

TABLE 1
An example profile.

Attribute Header	Attribute Value
Name	Jim Green
Sex	Male
Age	30
Home Town	New York City
University	Columbia
Profession	engineer
Interest	basketball, computer game
Location	40.758663,-73.981329

In this paper, we address the verifiable privacy preserving profile matching and secure communication channel establishment in decentralized MSN without any presetting or trusted third party. Formally, each user v_k in a social network has a profile set A_k consisting of m_k attributes, $A_k = \{a_k^1, a_k^2, ..., a_k^{m_k}\}$. Here the value a_k^i, the i-th dimension of the profile A_k, represents the i-th attribute of the user v_k. Each attribute consists of a header indicating its category (*e.g.*, "profession" and "interest") and a value field. The number of attributes is not necessary the same for different users. Table 1 shows an example profile. An initiator v_i represents his/her desired user by a request profile with m_t attributes as $A_t = \{a_t^1, a_t^2, ..., a_t^{m_t}\}$. Our mechanism allows the initiator to search a matching user in a flexible way by constructing the request profile in the form of $A_t = (N_t, O_t)$. Here

- N_t consists of α *necessary* attributes. All of them are required to be owned by a matching user;
- O_t consists of the rest $m_t - \alpha$ *optional* attributes. At least β of them should be owned by a matching user.

The acceptable *similarity threshold* of a matching user is

$$\theta = \frac{\alpha + \beta}{m_t}. \quad (1)$$

Let $\gamma = m_t - \alpha - \beta$. When $\gamma = 0$, a perfect match is required. A *matching user* v_m with a profile A_m must satisfy that

$$N_t \subset A_m \text{ and } |O_t \cap A_m| > \beta. \quad (2)$$

When $A_t \subset A_m$, v_m is a perfect matching user. In a decentralized MSN, a request will be spread by relays until hitting a matching user or meeting a stop condition, *e.g.* expiration time. Then the initiator v_i and the matching user v_m decide whether to connect each other.

2.2 Adversary Model

In the profile matching phase, if a party obtains one or more users (partial or full) attribute sets without their explicit consents, it is said to conduct *user profiling* [16]. Two types of user profiling are taken into consideration.

In the *honest-but-curious* (HBC) model, a user tries to learn more profile information than allowed by inferring from only the information he/she receives but honestly follow the mechanism. In a *malicious* model, an attacker tries to learn more profile information using background knowledge beyond his/her received data or by deliberately deviating from the mechanism. In this work we consider the following powerful malicious attacks.

Definition 1 (Dictionary profiling): A powerful attacker who has obtained the dictionary of all possible attributes tries to determine a specific user's attribute set by enumerating or guessing all combinations of attributes.

Definition 2 (Cheating): In the process of profile matching, a participant may cheat by deviating from the agreed protocol, *e.g.*, cheat the initiator with a wrong matching conclusion.

Most existing private profile matching approaches are vulnerable to the dictionary profiling attack and cheating.

In the communication phase, an adversary can learn the messages by eavesdropping. The construction of a secure channel may suffer the Man-in-the-Middle (MITM) attack.

There are also other saboteur attacks. In this work, we focus on addressing above attacks. For example, the deny of service (DoS) attack can be prevented by restricting the frequency of relaying requests from the same user. Some saboteur behaviors are precluded, such as inputting fake attributes, altering or dropping the requests or replies.

2.3 Design Goal

The main goal and great challenge of our mechanism is to conduct efficient matching against the user profiling and cheating, as well as establish a secure communication channel thwarting the MITM attack in a decentralized manner. In our mechanism, a user's privacy is protected from the user whose similarity is not up to his/her defined threshold. Specifically, we define different privacy protection levels $PPL(A_k, v_j)$ of a profile A_k of v_k against a user v_j.

Definition 3 (Privacy Protection Level): Four different privacy protection levels will be discussed in this work:

PPL0: If $PPL(A_k, v_j) = 0$, v_j can learn the profile A_k.

PPL1: If $PPL(A_k, v_j) = 1$, v_j can learn the intersection of A_k and A_j.

PPL2: If $PPL(A_k, v_j) = 2$, v_j can learn the α necessary attributes of A_k and the fact that at least β optional attributes are satisfied. Specially, when $\alpha = 0$, v_j learns the fact that the cardinality of $A_k \bigcap A_j$ exceeds the threshold.

PPL3: If $PPL(A_k, v_j) = 3$, v_j learns nothing about A_k.

We design our mechanism to achieve PPL2 against matching users and PPL3 against unmatched users in both HBC and malicious model and thwart cheating. In mobile social networks, our private profile matching and secure communication channel establishment mechanisms are also designed to be lightweight and practical, *i.e.*, low computation and communication cost, requiring no presetting, server and trusted third party. We also optimize the mechanism to reduce the overhead for unmatched users. Furthermore, in our mechanism human interactions are needed only to decide whether to connect their matching users.

3 Privacy Preserving Profile Matching and Secure Communication

Here we present our lightweight privacy preserving flexible profile matching in decentralized mobile social networks without any presetting or trust third party. A secure communication channel is simultaneously constructed between matching users. In fact, this mechanism can also be applied to other social networks without a trust server or services provider.

3.1 Basic Mechanism

Observe that the intersection of the request profile and the matching profile is a nature common secret shared by the initiator and the matching user. Our main idea is to use the request profile as a key to encrypt a message. Only a matching user, who shares the secret, can decrypt the message efficiently.

Figure 1 illustrates our basic privacy preserving search and secure channel establishment mechanism. Here, we first draw an outline of how the initiator creates a request and how a relay user handles the request.

The initiator starts the process by creating a *request profile* characterizing the matching user and a secret message containing a channel key for him/her. The request profile is a set of *sorted* attributes. Then he/she hashes the attributes of the request profile one by one to produce a *request profile vector*. A *profile key* is generated based on the request profile vector using some publicly known hashing function. The initiator encrypts the secret message with the profile key. A *remainder vector* of the profile vector is yield for fast exclusion by a large portion of unmatched persons. To support a flexible fuzzy search requiring no perfect match, the initiator can define the *necessary attributes*, *optional attributes* and the *similarity threshold* of the matching profile. And a *hint matrix* is constructed from the request profile vector according to the similarity definition, which enables the matching person to recover the profile key. In the end, the initiator packs the encrypted *message*, the *remainder vector* and the *hint matrix* into a *request package* and sends it out. Note that the required profile vector will not be sent out.

When a *relay user* receives a request from another user, he/she first processes a fast check of his/her own profile vector

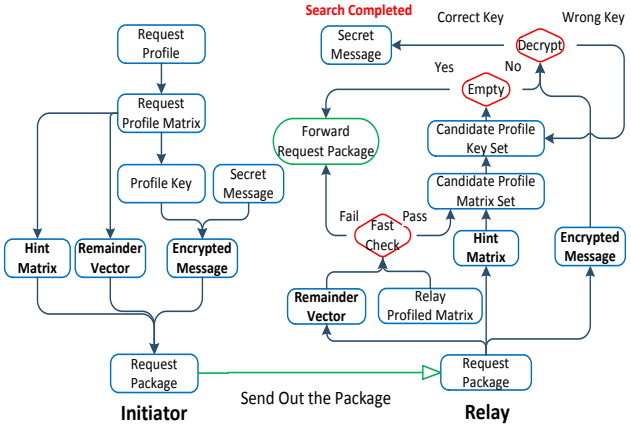


Fig. 1. The procedure to create a request package by the initiator and handle a request package by some forwarder.

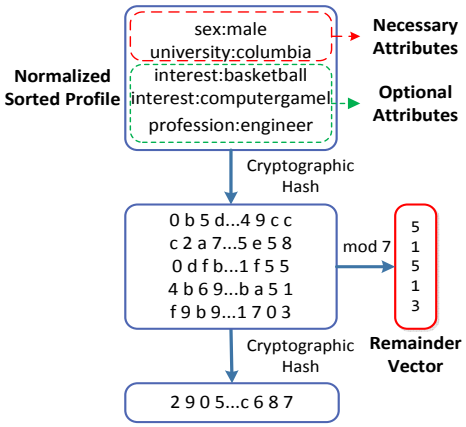


Fig. 2. The procedure to generate the *profile key* and remainder vector with a sample user profile.

with the remainder vector. If no sub-vector of his/her profile vector fits the remainder vector, he/she knows that he/she is unmatched and will forward the request to other relay users immediately. Otherwise, he/she is a *candidate* target and will generate a *candidate profile vector set* by some linear computation with his/her profile and the hint matrix. Then a *candidate profile key set* is obtained. In the basic mechanism, If any of his/her candidate keys can decrypt the message correctly, he/she is a matching user and the searching and secret key exchange complete. Otherwise, he/she just forwards the request to other relay users.

3.2 Profile Vector and Key Generation

To protect the profile privacy and support a fuzzy search, a cryptographic hash (*e.g.* SHA-256) of the attribute is adopted as the attribute equivalence criterion in this mechanism. However, due to the avalanche effect, even a small change in the input will result in a mostly different hash. Although consistent attribute header (category) can be provided by the social networking service, the attribute values are created by users. Some inconsistency may be caused by letter case, punctuation, spacing, *etc.*. For example, "basketball" and "Basketball" generate totally different cryptographic hashes. So a profile normalization is necessary before the cryptographic hashing to ensure two attributes which are considered equivalent to yield the same hash value. Words normalization has been well studied in research areas like search engines and corpus management [24]. In our mechanism, we use some common techniques to normalize the users profile, including removing whitespace, punctuation, accent marks and diacritics, converting all letters to lower case, converting numbers into words, text canonicalization, expanding abbreviations, converting the plural words to singular form. After the profile normalization, most inconsistences caused by spelling and typing are eliminated. The semantic equivalence between two different words are not in this paper's consideration. For some extremely noisy text or to achieve semantic equivalence, more sophisticated methods can be applied, *e.g.*, [24], [19] and [9].

Assume the cryptographic hash function is $\mathbf{H}$ which yields n-bit length hash value. With a *sorted* normalized profile $A_k = [a_k^1, a_k^2, ..., a_k^m]^T$, a *profile vector* is

$$H_k = \mathbf{H}(A_k) = [h_k^1, h_k^2, ..., h_k^m]^T. \quad (3)$$

Here $h_k^i = \mathbf{H}(a_k^i)$. A *profile key* is created with H_k,

$$K_k = \mathbf{H}(H_k). \quad (4)$$

Figure 2 shows the profile vector and key generation of an example profile in Table 1. With the key of the required profile, the initiator encrypts the secret message using a symmetric encryption technique like Advanced Encryption Standard (AES). Any person who receives it tries to decrypt the secret message with his/her own profile key. Only the exactly matching person will decrypt the message correctly.

3.3 Remainder Vector and Hint Matrix

So far, with the profile key, we have realized a naive private profile matching and secure channel establishment. However, the naive mechanism has some flaws making it unpractical.

1) The search is not *flexible*. The initiator cannot query any subset of other's profile. For example, he/she need to find a "student" studying "computer science" regardless of the "college".
2) A perfect matching is required and *no fuzzy* search is supported. In most cases, the initiator need only find some person with profile exceeding the required similarity threshold to the requested profile.
3) All participants must decrypt the message, although most of them hold wrong keys. It wastes the computation resource and increases the search delay.

Improving the naive basic mechanism, our new mechanism allows the initiator to search a user in a flexible way $A_t = (N_t, O_t)$, as described in Section 2.1. We use a *Remainder Vector* for fast excluding most unmatched users. And a *hint matrix* is designed to work with the remainder vector to achieve efficient free-form fuzzy search.

3.3.1 Remainder Vector

Assume that there are m_t attributes in the request profile, p is a prime larger than m_t. A *remainder vector* R_k consists of the remainders of all hashed attributes in the input H_k divided by p, as illustrated in Fig. 2.

$$R_k = [h_k^1 \mod p, h_k^2 \mod p, ..., h_k^m \mod p]^T. \quad (5)$$

Then the following theorem is straightforward.

Theorem 1: Consider two attributes' hashes $h^i = \mathbf{H}(a^i)$ and $h^j = \mathbf{H}(a^j)$, remainder $r^i \equiv h^i \mod p$ and remainder $r^j \equiv h^j \mod p$. If $r^i \neq r^j$, then $h^i \neq h^j$.

Assume that the remainder vector of the required profile A_t is $R_t = [r_t^1, r_t^2, ..., r_t^{m_t}]^T$, and a relay user's profile vector is H_k. To use the remainder vector R_t to conduct a fast check, a relay user should firstly calculates m_t *candidate attribute subsets* $H_k(r_t^i)$ fitting each r_t^i. Here $\forall h_k^x \in H_k(r_t^i) : h_k^x \mod p = r_t^i$, *i.e.*, attributes in $H_k(r_t^i)$ yields the same remainder r_t^i when divided by p.

Secondly, a combination of one element from each candidate attribute subset forms a profile vector of the relay user. If the candidate attribute subset $H_k(r_t^i)$ is $\emptyset$, the corresponding element in the combination will be set to *unknown*, which means that the relay user fails to meet the required attribute a_t^i according to Theorem 1.

Thirdly, the relay user checks if he/she is a *candidate matching user* of the request by determining that if there exist at least one combination, denoted by H_c, which satisfies the following:

1) The α necessary attributes are all known, *i.e.*

$$H_k(r_t^i) \neq \emptyset, \forall i \leq \alpha; \quad (6)$$

2) The number of unknown elements don't exceed γ, *i.e.*

$$|\{H_k(r_t^i) \mid \alpha < i \leq m_t,\ H_k(r_t^i) = \emptyset\}| \leq \gamma; \quad (7)$$

3) Since H_t and H_k are both sorted, the elements of H_c should still keep the order consistent with H_k, *i.e.*

$$\begin{aligned} &\forall h_k^x \in H_c, h_k^y \in H_c. \\ &h_k^x \in H_k(r_t^i), h_k^y \in H_k(r_t^j), i < j \Rightarrow x < y. \end{aligned} \quad (8)$$

We call H_c a *candidate profile vector*. Without satisfying the three conditions, a profile is not possible to match the request and will be excluded immediately.

In a fuzzy search, during the fast checking procedure, if there is no candidate profile vector that can be constructed by the relay user's profile vector, then he/she is unmatched and he/she can forward the package. An ordinary relay user commonly has only dozens of attributes, so there won't be many candidate profile vectors. Using the remainder vector, quick exclusions of a portion of unmatched users can be made.

3.3.2 Hint Matrix

A *hint matrix* is constructed to support a flexible fuzzy search. It describes the linear constrain relationship among the $\beta + \gamma$ optional attributes. With its help a matching user exceeding the similarity threshold can recover γ unknown attributes, so as to generate the correct profile key. Note that when a perfect matching user is required, no hint matrix is needed.

The *constrain matrix* with γ rows and $\gamma + \beta$ columns is:

$$C_{\gamma \times (\gamma+\beta)} = [I_{\gamma \times \gamma}, R_{\gamma \times \beta}]. \quad (9)$$

Here matrix I is a γ dimensional identity matrix, R is a matrix of size $\gamma \times \beta$, each of its elements is a random nonzero integer.

Multiplying the constrain matrix to the optional attributes of the required profile vector, the initiator gets a matrix B:

$$B = C \times [h_t^{\alpha+1}, h_t^{\alpha+2}, ..., h_t^{m_t}]^T \quad (10)$$

Then the *hint matrix* M is defined as matrix C, followed by matrix B, *i.e.*,

$$M = [C, B]. \quad (11)$$

When $\gamma > 0$, the initiator generates the hint matrix and sends it with the encrypt message and the remainder vector.

In a fuzzy search, after the fast check, if the relay user is a candidate matching user, he/she constructs a set of candidate profile vector H_c with unknowns. By definition of the candidate profile vector, each H_c has no more than γ unknowns, and any unknown h_c^i, which is the i-th element of H_c, has $i > \alpha$. Now, the unknowns of a candidate profile vector can be calculated by solving a system of linear equations:

$$C \times [h_c^{\alpha+1}, h_c^{\alpha+2}, ..., h_c^{m_t}]^T = B \quad (12)$$

Equivalently, we have

$$[I, R] \times [h_c^{\alpha+1}, h_c^{\alpha+2}, ..., h_c^{m_t}]^T = B. \quad (13)$$

This system of equations has equal to or less than γ unknowns. It has a unique solution. With the solution, a complete candidate profile vector H_c' is recovered. For each H_c', a candidate key $K_c = \mathbf{H}(H_c')$ can be generated. If any of the relay user's candidate keys decrypts the message correctly, he/she is a matching user and gets the encrypted secret. Else, he/she forwards the request to the next user.

3.4 Location Attribute and Its Privacy Protection

In localization enabled MSN, a user usually searches matching users in vicinity. Most systems require a user to reveal his/her own location, which violates the user's privacy [18]. There are some work dedicated to privacy-preserving proximity discovering, *e.g.*, Sharp [17], [36]. Typically, a vicinity search is defined as follows: given a user v_i's current location $l_i(t)$ at time t and his/her vicinity range D, if a user v_k is in his vicinity then $distance(l_i(t), l_k(t)) \leq D$ should be satisfied. In our mechanism, we consider location as a dynamic attribute which will be updated while the user moves, and design a location privacy preserving vicinity search method compatible with our private profile matching mechanism by converting the distance measuring to location attribute matching. To generate the location attributes, we use lattice based hashing to hash a user's current location (*e.g.*, GPS coordinates) and his/her vicinity region. Then a private vicinity search can be easily conducted via our fuzzy search scheme with the help of the hint matrix. The dynamic attribute also improves the privacy protection for static attributes.

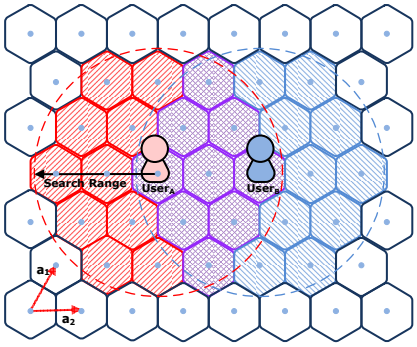


Fig. 3. Lattice-based location hash.

3.4.1 Lattice based Location Hashing

Our system maps the two-dimensional location to the hexagonal lattice. The lattice point set is a discrete set of the centers of all regular hexagons, as the dots shown in Fig. 3. The lattice is formally defined as

$$\{x = u_1 a_1 + u_2 a_2 \mid (u_1, u_2) \in \mathbb{Z}^2\} \tag{14}$$

Here a_1, a_2 are linearly independent primitive vectors which span the lattice. Given the primitive vectors a_1, a_2, a point of the lattice is uniquely identified by the integer vector $u = (u_1, u_2)$. There are infinite choices for a_1, a_2. Let d denote the shortest distance between lattice points, for simplicity, we choose the primitive vectors as presented in Figure 3:

$$a_1 = (d, 0); a_2 = (\frac{1}{2}d, \frac{\sqrt{3}}{2}d). \tag{15}$$

Given a geography location as the origin point O and the scale of the lattice cell d by the application, with the lattice definition, a location can be hashed to its nearest lattice point. Any two locations hashed to the same lattice point are inside a single hexagonal lattice cell, and they are separated by a bounded distance d. Then a user v_k's current vicinity region can be hashed to a *lattice point set*, $V_k(O, d, l_k(t), D)$, consisting of lattice points, i.e. the hash of $l_k(t)$, and other lattices points whose distances to the center lattice point are less than the range D.

3.4.2 Location Privacy Preserved Vicinity Search

Intuitively, given the distance bound to define vicinity, if two users v_i and v_k are within each other's vicinity, the intersection of their vicinity regions V_i and V_k will have a proportion no less than a threshold Θ. Here Θ is determined by $\min \frac{area(V_i \bigcap V_k)}{area(V_i)}$ when $distance(l_i(t), l_k(t)) \leq D$. Typically, if the vicinity region is a circle, $\Theta = 0.4$. In our mechanism, the initiator calculates his/her vicinity lattice point set $V_i(O, d, l_i(t), D)$. If a user v_k is in his/her vicinity, then v_k's vicinity lattice point set $V_k(O, d, l_k(t), D)$ should satisfy the requirement:

$$\theta_k = \frac{|V_i(O, d, l_i(t), D) \bigcap V_k(O, d, l_k(t), D)|}{|V_k(O, d, l_k(t), D)|} \geq \Theta \tag{16}$$

In the example in Fig. 3, user v_A's vicinity range $D = 3d$. The lattice points within the red slashed hexagons constitute $V_A(O, d, l_A(t), D)$ and the lattice points within the blue back slashed hexagons constitute $V_B(O, d, l_B(t), D)$. The purple grid region is the intersection of vicinity regions of v_A and v_B. In this case, $\theta_B = \frac{9}{19}$, which is larger than Θ (0.4), hence v_B is in the vicinity of v_A. When v_B moves out of the vicinity of v_A, θ_B will become smaller than Θ. In practice, the hash function (*i.e.*, the lattice) is defined by the application and shared by all devices. The initiator can adjust the parameter D to change the cardinality of the vicinity lattice point set to a suitable size.

To conduct location privacy preserved vicinity search, the initiator won't send his/her vicinity lattice point set directly. Using the sorted lattice points, he/she generates a dynamic profile key, a dynamic remainder vector and a dynamic hint matrix in the same way as processing other attributes. So a vicinity search works as a fuzzy search with similarity threshold Θ. Only participants in his/her vicinity who has a certain amount of common lattice points with him/her can generate the correct dynamic profile key with the help of the dynamic remainder vector and hint matrix.

Protocol 1: Privacy Preserving Profile Matching

1) The initiator encrypts a random number x and a public predefined confirmation information in the secret message $E_{K_t}(confirmation, x)$ by the required profile key K_t. And he/she sends the request out.
2) A candidate relay user can verify whether he/she decrypts the message correctly by the confirmation information. If he/she does not match, he/she just forwards the request to the next user. If he/she is matching, he/she can reply the request by encrypting the predefined ack information and a random number y along with any other message (e.g. the intersection cardinality) by x, say $E_x(ack, y)$, and sends it back to the initiator.

3.4.3 Location Based Profile Matching

Compared to static attributes like identity information, location is usually a temporal privacy. With the *dynamic key* generated from location information, we can improve the protection of a user's static attributes. When constructing profile vector of a user, we can hash the concatenation of each static attribute and his/her current dynamic key instead of directly hash static attributes. So the hash values of the same static attribute are completely changed when user update his/her location. It will greatly increase the difficulty for the malicious adversary to conduct dictionary profiling. However, it won't bring much more computation for ordinary user because of quite limited lattice points in his/her vicinity range.

3.5 Privacy-Preserving Profile Matching Protocols

We are now ready to present our privacy preserving profile matching protocols. Here we present three different protocols that achieve different levels of privacy protection.

3.5.1 Protocol 1

Under Protocol 1, an unmatched user doesn't know anything about the request. The matching user knows the intersection

Protocol 2: Privacy Preserving Profile Matching

1) The initiator encrypts a random number x in the secret message $E_{K_t}(x)$ by the request profile key K_t.
2) A candidate matching user cannot verify whether he/she decrypts the message correctly. Let the candidate profile key set be $\{K_c^1, K_c^2, \ldots, K_c^z\}$. He/she decrypts the message in the request with each candidate profile key to get a set of numbers, say $U = \{u_j \mid u_j = D_{K_c^j}(E_{K_t}(x))\}$. Then he/she encrypts the predefined ack information and a random number y by each u_j as the key, and sends the acknowledge set $\{E_{u_j}(ack, y)\}$ to the initiator, for a public ack.
3) The initiator excludes the potential malicious repliers whose response time exceeds the time window or the cardinality of reply set exceed the threshold. He/she decrypts the replies with x. If he/she gets a correct ack information from a reply, the corresponding replier is a matching user.

of the required profile and his/her own profile after Step 1 in the HBC model, and he/she can decide whether to reply. The initiator doesn't know anything about any participant until he/she gets a reply. With replies, he/she knows the matching users and even the most similar replier by the cardinality information. Then he/she can start secure communication with a matching user encrypted by $x+y$ or with a group of matching users encrypted by x. However, in malicious model, if the matching user has a dictionary, he/she can learn the whole request profile by the recovered profile vector.

3.5.2 Protocol 2

To prevent malicious participants, we design Protocol 2, which is similar to Protocol 1, but it excludes the confirmation information from the encrypted message.

Under Protocol 2, after the first round of communication, the participants won't know anything about the request in both HBC model and malicious model. The initiator knows who are the matching users and even the most similar one according to the replies. Then the initiator can start secure communication with a matching user protected by the key $x+y$ or with a group of matching users protected by x. In malicious model, if a participant has a dictionary of the attributes, he/she may construct a large candidate profile key set and send it to the initiator. The main difference between an ordinary user and a malicious user with a dictionary is the size of their attribute space. However, an ordinary user with about dozens of attributes can make a quick reaction and reply a small size acknowledge set. While it takes much longer for a malicious user due to a large number of candidate attribute combinations. So the initiator can identify the malicious repliers by response time and the cardinality of reply set. In practice, the response time window should be less than 1 second based on the experiments with our mobile social networking system WiFace [31]. According to our real data based evaluation, the threshold of cardinality could be set to 12, as shown in Fig. 8.

Protocol 3: User-defined Privacy Preserving Profile Matching

1) The initiator encrypts a random number x in the secret message without any confirmation information by the required profile key, say $E_{K_t}(x)$.
2) A candidate matching user cannot verify whether he/she decrypts the message correctly. He/she selects a set of candidate profile $\{A_c^1, A_c^2, \ldots, A_c^z\}$ which satisfies that $S(\bigcup_{i=1}^{z} A_c^i) \leq \varphi$. And he/she generates the corresponding candidate profile keys $\{K_c^1, K_c^2, \ldots, K_c^z\}$. He/she decrypts the message in the request with each candidate profile key to get a set of numbers, say $U = \{u_j \mid u_j = D_{K_c^j}(E_{K_t}(x))\}$. Then he/she encrypts the predefined ack information and a random number y by each u_j, and sends back the acknowledge set $\{E_{u_j}(ack, y)\}$ back to the initiator.
3) The initiator excludes the malicious replier whose response time exceeds the time window or the cardinality of reply set exceed the threshold and decrypts the replies with x. If he/she gets a correct ack information from a reply, the corresponding replier is matching.

Using our profile matching mechanism, an adversary cannot build an attribute dictionary in our social networking system. If an attribute dictionary is constructed using other sources, *e.g.*, other similar social networking systems, the space of attributes are mostly very large, which makes the dictionary profiling infeasible. Especially in the localizable social mobile social networks, the vast dynamic location attribute will greatly increase the difficulty of dictionary profiling. However, there still may exist a special case that the attribute space is not large enough in some social networks. Consider an unlikely case that, an adversary constructs the attribute dictionary from other similar social networking system and the the attribute space is not large enough. In this case, in Protocol 1, the request profile may be exposed via dictionary profiling by malicious participants. Although Protocol 2 protects the request profile from any participants, a malicious initiator may learn the profile of unmatching repliers.

3.5.3 Protocol 3

To prevent the dictionary profiling by malicious initiator, we improve Protocol 2 to Protocol 3 which provides a user personal defined privacy protection. Here, we use attribute entropy to represent the amount of privacy information contained in an attribute, also the severity of the attribute value leakage.

Definition 4 (Attribute Entropy): For an attribute a^i with t^i values $\{x_j : j = 1, \ldots, t^i\}$. $P(a^i = x_j)$ is the probability that the attribute a^i of a user equals x_j. The entropy of the attribute a^i is

$$S(a^i) = -\sum_{j=1}^{t^i} P(a^i = x_j) \log P(a^i = x_j). \quad (17)$$

Definition 5 (Profile Entropy): The entropy of profile A_k is

$$S(A_k) = \sum_{i=1}^{m_k} S(a^i). \quad (18)$$

Especially, when the t^i values of a^i have equal probability, the entropy of the attribute a^i is $S(a^i) = \log t^i$. In this case the entropy of the profile A_k is $S(A_k) = \log \prod_{i=1}^{m_k} t^i$.

Intuitively, the larger the entropy of a profile, the more severe the leakage of the profile value. Taking attributes "gender" and "birthday" as example, "gender" has smaller entropy and less privacy-sensitive, while "birth day" has larger entropy and is usually more privacy-sensitive. So, the leakage of "birthday" value is more severe than the leakage of "gender" value. A participant can determine his/her personal privacy protection level by giving an acceptable upper limit φ of profile leakage, which is measure by the entropy of the leaked profile. Based on the user defined protection level, Protocol 3 is φ-entropy private for each user.

Definition 6 (φ-Entropy Private): A protocol is φ-entropy private when the entropy of leaked profile is not greater than the upper limit φ:

$$S(Leak(A_k)) \leq \varphi. \tag{19}$$

Here the $Leak(A_k)$ indicates the set of leaked attributes of profile A_k.

A user can use k-anonymity (thus $\varphi = \log \frac{n}{k}$) or use the most sensitive attribute (thus $\varphi = \min(S(a^i))$) to decide φ. The parameter φ is decided by each user. Here we suggest two options to determine φ.

(1) *K-anonymity based.* To prevent from being identified by disclosed attributes, the user will only send out messages protected by the profile key generated from the subset of attributes which at least k users own the same subset. Let a subset of A_k be $A_k^s = a^1, \ldots, a^l$. Suppose that the t^i values of a^i have equal probability, then there are $\frac{n}{\prod_{i=1}^{l} t^i}$ users who are expected to own the same subset. If the user require that $\frac{n}{\prod_{i=1}^{l} t^i} \geq k$, then $\log \prod_{i=1}^{l} t^i \leq \log \frac{n}{k}$, ie $S(A_k^s) \leq \log \frac{n}{k}$. So the parameter φ could be $\log \frac{n}{k}$. In this way, we can approximate k-anonymity privacy.

(2) *Sensitive attributes based.* The user can determine the sensitive attributes which must not be disclosed according to the current context. Let the set of sensitive attributes defined by user v_k is A_k^s, then $\varphi = \min(S(a^i))$, where $a^i \in A_k^s$.

Protocol 3 is privacy-preserving when the initiator is not malicious. and it is φ-private for each participant even when the initiator can conduct a dictionary profiling.

Note that, for all three protocols, each request has a valid time. An expired request will be dropped. And each user (identified by his/her network ID) has a request frequency limit, all participants won't reply the request from the same user within a short time interval. Other saboteur attacks, for example DoS attack with fake identities and inputting fake attributes, could be prevented using some existing methods, and they are out of scope of this work.

3.6 Establishing Secure Communication Channel

As presented in the profile matching protocols, the random number x generated by the initiator and y generated by a matching user have been exchanged secretly between them, which is resistant to the Man-in-the-Middle attack from any unmatched users. A pair of matching users can use $Hash(x \oplus y)$ as their communication key. Furthermore, our mechanism also discovers the community consisting of users with similar profile as the initiator and establish the group key x for secure intra-community communication.

TABLE 2
The privacy protection level (PPL) of our protocols. v_i is the initiator, v_m is a matching user and v_u is an unmatching user. A_i, A_m and A_u are their corresponding profiles. v_i' is a malicious initiator with a profile dictionary. v_p' is a malicious participant with a profile dictionary eavesdropping the communication. NC stands for non-candidate, and CD stands for candidate. PPL is defined in Definition 3.

(a) In HBC model.

PPL	(A_I, v_M)	(A_I, v_U)	(A_M, v_I)	(A_U, v_I)
Protocol 1	1	3	2	3
Protocol 2	3	3	2	3
Protocol 3	3	3	2	3
PSI	3	3	1	1
PCSI	3	3	$\lvert A_I \cap v_U \rvert$	$\lvert A_I \cap v_U \rvert$

(b) In Malicious model.

PPL	Protocol 1	Protocol 2	Protocol 3
(A_I, v_P')	0	3	3
(A_M, v_I')	2	2	φ-entropy
(A_M, v_P')	2	3	3
(A_U, v_I')	3	3 for NC 2 for CD	3 for NC φ-entropy for CD
(A_U, v_P')	3	3	3

4 Security and Efficiency Analysis

In this section, we analyze the security and performance of our profile matching and secure communication channel establishing mechanism.

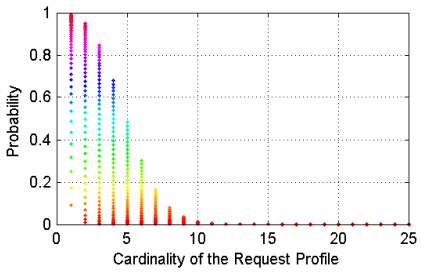
4.1 Security and Privacy Analysis

4.1.1 Profile Privacy

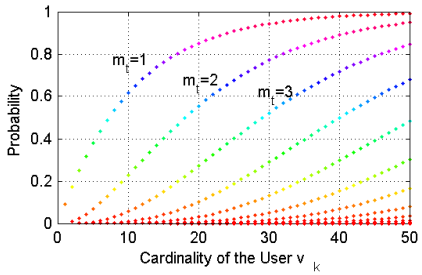
During the generation of the profile key, we use the hash value (with SHA-256) of the combination of the static attribute and the dynamic attribute (*i.e.* location). The SHA-256 is collisions and preimage resistance without any successful existing attack. Moreover, the dynamic attribure greatly increases the protection of the static attribute. We use 256-bit-key AES as the encryption method, which is infeasible to break directly. The 256-bit profile key is used as the secret key to encrypt the message by AES. Only the encrypted message will be transmitted, and no attribute information will be transmitted in any data packets. Therefore no user can obtain other user's attribute hash to build a dictionary. To acquire the profile information of the initiator or other participants the attacker needs to decrypt the request/reply message correctly and confirm the correctness. This is extremely difficult due to the choice of SHA-256, 256-bit-key AES and the random salt x.

In the HBC model only users owning matching attributes can decrypt each other's messages correctly. Unmatched user cannot obtain any information from the encrypted message. Table 2(a) presents the protection level of our protocols, which

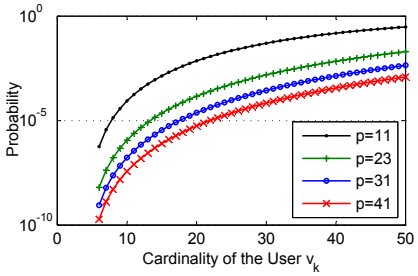
Fig. 4. Probability that v_k's profile matrix contains a candidate profile matrix.



(a)Probability changing with the cardinality of a request profile ($p = 11$).



(b)Probability changing with the cardinality of a participant's profile ($p = 11$).



(c)Probability changing with different prime number ($m_t = 6$).

is defined in Definition 3, in the HBC model. Compared to the existing PSI and PCSI approaches, our protocols provide Level 2 privacy protection for matching users and don't leak any information to unmatched users.

In the malicious model, the adversary cannot build an attribute dictionary with our system. But we still consider the situation that an adversary constructs the dictionary from other sources, *e.g.*, other similar social networking systems. In practice the attribute could be the user's profession, interest, race, favorite song, etc. So in most cases, the cardinality m of the dictionary is very large, which makes the dictionary profiling difficult. With a remainder vector, it takes an adversary $(\frac{m}{p})^{m_t}$ guesses to compromise a user's profile with m_t attributes. Here p is a small prime number like 11, while for most social networking systems m is very large. For example, in Tencent Weibo, we found that $m \simeq 2^{20}$ and the average attribute number of each user is 6. When the adversary tries to guess a user's profile by brute force, it will take about 2^{100} guesses. If considering keywords of a user, m is even much larger. Especially in localizable MSN, the vast dynamic location attribute will greatly increase the attribute set and make the dictionary profiling more infeasible. There is an unlikely case that the dictionary size is not large. Table 2(b) shows the protection level of our protocols in this case. Protocol 1 cannot protect the initiator's privacy from the dictionary profiling. But it provides Level 2 privacy for replying matching users and Level 3 privacy for other users. Protocol 2 provides Level 3 privacy for the initiator. It also provides Level 3 privacy for all participants against any other persons except the initiator. Only if the initiator is an adversary with the dictionary, he/she may compromise the candidate user's privacy. Protocol 3 still provides Level 3 privacy for the initiator and incardinate users, and Level 3 privacy for the candidate users against any other person except the initiator. Moreover, it provides personal defined φ-entropy privacy for all candidate users against a malicious initiator.

4.1.2 Communication Security

Our protocols realize secure key exchange between matching users based on their common attributes. The shared secret key is protected by the profile key, only the user who owns the matching attributes can generate the same profile keys. As the security analysis of our protocols, Protocol 2 and 3 can construct a secure communication between a matching user and the initiator against any other unmatched adversary in both HBC and malicious model. So our secure communication channel establishment between matching users thwarts the MITM attack from any unmatched users.

4.1.3 Verifiability

In majority of existing profile matching approaches, only one party learns the true result, and he/she tells the other party the result. There lacks an efficient way for the other party to verify the result. Our protocols are *verifiable* and resists cheating. In our protocols, matching users are required to reply $E_x(ack, y)$. In the HBC model, only the matching user can get the correct x. An unmatched user cannot cheat the initiator to pretend to be matched. Consequently, the initiator can only obtain the correct y of the matching user. So both the initiator and participants cannot cheat each other. In the malicious model, it takes an adversary with a dictionary much longer time to guess the correct key. Hence a nonmalicious user can distinguish the adversary by his/her reply delay. So no cheating can be conducted successfully with our protocols.

4.1.4 Other Attacks

There are other saboteur attacks. The case that the initiator sends requests frequently to profile the network or conduct the deny of service (DoS) attack can be prevented by restricting the frequency of relaying and replying requests from the same user. Note that, some saboteur behaviors are not in the concern of this work, such as alter or drop the requests or replies.

4.2 Performance Analysis

Theorem 2: The expected number of candidate profile vector of H_k is

$$\binom{m_k}{\alpha+\beta} \times \left(\frac{1}{p}\right)^{\alpha+\beta} = \frac{(\alpha+\beta)!}{(\alpha+\beta)!(m_k-\alpha-\beta)!} \times \left(\frac{1}{p}\right)^{m_t}. \quad (20)$$

Proof: Omitted due to space limitation. □

Theorem 3: The cardinality of the request profile vector H_t is m_t. The cardinality of the profile vector H_k is m_k. The probability that H_k contains a candidate profile vector which

satisfies the order constraint and yields the same remainder vector as H_t is

$$\rho_{m_t m_k} = \omega^{m_t}\Big(\sum_{i=1}^{m_k-m_t}\binom{i+m_t-1}{i}(1-\omega)^i\Big), \quad m_t \geq 1, m_k \geq m_t. \tag{21}$$

Proof: Omitted due to space limitation. □

4.2.1 Computational Cost

For an initiator, it takes $O(m_t \log m_t)$ operations for sorting attributes, m_t+1 hashing operations for profile key generation and m_t modulo operations for remainder vector generation. $\gamma(\gamma+\beta)$ operations are needed to calculate the hint matrix if the required similarity $\theta < 100\%$. One symmetric encryption is needed with the profile key.

For a participant v_k, it takes $O(m_k \log m_k)$ operations for sorting its attributes, m_k hashing for profile vector generation and m_k modulo operations for remainder calculation. After fast checking by remainder vector, if a participant v_k is not a candidate user, no more computation is need. If v_k is a candidate user, let the number of his candidate profile vector be κ_k. It takes this user κ_k hashing to generate candidate profile keys. If there is a hint matrix, user v_k needs to solve κ_k $\gamma(\gamma+\beta)$-dimension linear equation systems. The computation cost is $O(\kappa_k m_t^3)$. In the end, κ_k symmetric decryptions with the profile key. According to Theorem 2, the expected number of the candidate profile key is $\varepsilon(\kappa_k) = \binom{m_k}{\alpha+\beta} \times \left(\frac{1}{p}\right)^{\alpha+\beta}$. Although the whole attribute set of a social networking system is large, the attribute number of each user usually is very small. For example, in Tencent Weibo the average attribute number is 6 and the maximum number is 20. Then if $\alpha+\beta = 6$, even a large $m_k = 20$ and small prime number $p = 11$ result in a very small $\varepsilon(\kappa_k) = 0.02$. So it takes very small computation cost even for a candidate user. Based on Theorem 3, Figure 4 presents the number of candidate users. It shows that the expected candidate users is only a small portion of all users and the portion decreases greatly with the increase of m_t and p. In fact, there is a tradeoff behind the choice of p. Increasing p could reduce the cost of non-malicious users but it may be considered that larger p will weaken the security due to the decreased difficulty of dictionary profiling. Our testing and analysis show that even a small p, *e.g.*, $p = 11$, can significantly reduce the number of candidate users to a very small one, as shown in Figure 4, with little hurt to the security. So an initiator can easily choose a small p to efficiently control the amount of candidate users as well as achieve the secure protection of profile privacy.

In implementation, we use SHA-256 as the hashing method and AES as the symmetric encryption method. The data processing performance of SHA-256 is 111MB/s for a single-threaded implementation on an Intel 1.83GHz processor in 32-bit operation system. Furthermore, for all users the sorting and hashing results are calculated once and used repetitively until the attributes are updated, *e.g.*, the location changes. AES performs well on a wide variety of hardware, from 8-bit smart cards to high-performance computers. The throughput of AES is about 60MB/s on a 1.7GHz processor and about 400MB/s on Intel i5/i7 CPUs. Compared to the existing approaches, we don't use an asymmetric-key cryptosystem and the remainder vector can significantly reduce the computation of unmatching users. In all, our protocol are computationally efficient.

4.2.2 Communication Cost

In our protocols there isn't any pre-operation for key exchange or secure channel construction. To conduct profile matching with all users, the initiator only need one broadcast to send the request to all participants. The request consists of a $32m_t$ bit remainder vector and a 256 bit encrypted message. If the required similarity $\theta < 100\%$, there is also a $32\gamma(\gamma+\beta)+256\gamma$ bit hint matrix. The size of the request message is at most $(1-\theta)32m_t^2 + (288-256\theta)m_t + 256$ bits. For example, if a user has 6 attributes in average and 20 attributes at most. To search a 60% similar user, the request is about 190B in average and 1KB at most. In Protocol 1, only the matching user will reply the request. So the transmission cost of Protocol 1 is one broadcast and $O(1)$ unicasts. In Protocol 2, only the candidate matching user will reply the request. So the transmission cost of Protocol 2 is one broadcast and $O(n * (\frac{1}{p})^{m_t\theta})$ unicasts. For example, when $p = 11$, $m_t = 6$, $\theta = 0.6$, there are only about $\frac{1}{5610}$ fraction of users will reply. And as analyzed in Section 4.2.1, for an ordinary candidate the expected number of candidate profile keys is very small, hence the communication overhead is small too (the mean cost is less than 256 bit). In Protocol 3, the communication cost of reply is even smaller than Protocol 2 because of the personal privacy setting. Note that a reply in all three protocols is only 32Byte. So the communication cost of our protocols is quite small. This makes our protocols suitable for wireless communication environment, for example the mobile social networks.

4.2.3 Further Comparison With Related Work

We then compare the computation cost and communication cost between our protocol and PSI and PCSI based approaches. The computation time comparison with dot product based approach presented in [6].

First we define the basic operations of other asymmetric cryptosystem based approaches:

- M_{24}: 24-bit modular multiplication;
- M_{1024}: 1024-bit modular multiplication;
- M_{2048}: 2048-bit modular multiplication;
- E_{1024}: 1024-bit exponentiation;
- E_{2048}: 2048-bit exponentiation.

Then we define the basic operation in our protocol:

- $\mathcal{H}$: one SHA-256 hashing operation of an attribute;
- $\mathcal{M}$: is the operation that the 256-bit hash of an attribute modulo the small prime p;
- $\mathcal{E}$: AES-256 encryption;
- $\mathcal{D}$: AES-256 decryption.

Table 3 presents the computation and communication cost of related work and our protocol. In this table, we omit the performance of Protocol 2 and Protocol 3 due to the space limitation, and they have very similar performance as Protocol 1. Because the SHA-256, modulo, and AES-256 operations in

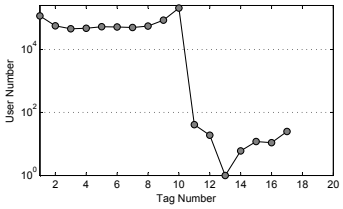


Fig. 5. Users' attribute number distribution.

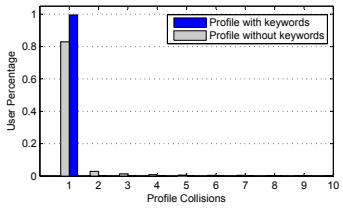


Fig. 6. Profile uniqueness and collisions of users.

our protocol are much cheaper than the 1024-bit and 2048-bit modular multiplication and exponentiation, our protocol costs much less computation than asymmetric-key based schemes. The transmitted data size are significantly reduced because basically only one encrypted 256-bit message and one $32 * m_t$ bit remainder vector need to be transmitted. Furthermore, the remainder vector eliminates the reply from most unmatching users. So the total transmission time is also reduced.

5 Real Data and System Evaluation

5.1 Real Social Networking System Analysis

To evaluate the practicality of mechanism, we use the profile data of a real-life social networking system, Tencent Weibo [2]. Tencent Weibo is one of the largest social networking platforms for building friendship and sharing interests online. This dataset has 2.32 million users' personal profiles. And the attributes include their ID, year of birth, the gender, the tags and keywords. Tags are selected by users to represent their interests. If a user likes mountain climbing and swimming, he/she may select "mountain climbing" or "swimming" to be his/her tag. Keywords are extracted from the tweet/retweet/comment of a user and can be used as attributes to better represent the user. There are total 560419 tags and 713747 keywords. As presented in Figure 5, each user has 6 tags in average and 20 tags at most. On average, 7 keywords extracted for a user. So when the adversary tries to guess a user's profile with 6 tags by brute force, it will take him/her about 2^{100} guesses. Moreover, the large attribute space makes the vector-based matching approaches impractical. Each profile is a combination of several attributes. When more than one user has the same profile, we say there are collisions for this profile. Figure 6 shows that more than 90% users have unique profiles.

5.2 Computation and Communication Performance

We then exam the computation performance of our protocols on mobile devices and PC. The mobile phone is HTC G17 with $1228Hz$ CPU, 1GB RAM. The laptop is Think Pad X1 with i7 2.7GHz CPU and 4GB RAM. Table 4 shows the mean execution time of our basic computation and Table 5 shows the mean execution time for basic operations for asymmetric cryptosystem based scheme. The basic operations of our mechanism are much cheaper than the 1024-bit or 2048-bit modular multiplication and exponentiation used by the asymmetric cryptosystem based schemes, *e.g.* the evaluation result of [6]. We evaluate our protocol on the dataset. Table 6 presents the breakdown of time cost for our protocol. As an example, for a user with 6 attributes, the time need to generate a request is only about 3.9×10^{-2} ms on laptop and 1.3 ms on mobile phone. On average it takes a non-candidate user about 3.9×10^{-2} ms on laptop and 0.63 ms on phone to process the request. For a candidate user the computation cost is about 4×10^{-2} ms on laptop and 7 ms on phone for each candidate key. Even when the attribute number of user increases to 1000, our protocols requires only about 1 s for him/her to process a request. Table 7 shows a typical scenario in a mobile social network with 100 users. With the comparison of numerical result based on implementation on the mobile phone, it is clearly that our protocol is much more efficient in both computation and communication.

TABLE 4
Mean computation time for our basic operation.(ms)

	$\mathcal{H}$	$\mathcal{M}$	$\mathcal{E}$
Laptop	1.2×10^{-3}	3.1×10^{-4}	8.7×10^{-4}
Phone	4.8×10^{-2}	5.7×10^{-2}	2.1×10^{-2}
	Multiply-256	Compare-256	$\mathcal{D}$
Laptop	1.4×10^{-4}	1.0×10^{-5}	9.6×10^{-4}
Phone	3.2×10^{-2}	1.0×10^{-3}	2.5×10^{-2}

TABLE 5
Mean computation time for basic operations for asymmetric cryptosystem based scheme.(ms)

	E_{1024}	E_{2048}	M_{1024}	M_{2048}
Laptop	17	120	2.3×10^{-2}	1×10^{-1}
Phone	34	197	1.5×10^{-1}	2.4×10^{-1}

We also investigate the communication delay on our Wi-Fi based multihop social networking system named WiFace [31]. 50 heterogeneous mobile nodes participated in WiFace to conduct friend matching and the average path length is about 3 hops. The average communication delay for a query is about 220 ms, and the largest delay is less than 800 ms.

5.3 Protocol Performance Evaluations

We evaluate the efficiency of our protocols with two typical situations. In the first case, all users have equal size of attributes which is similar to the vector based scheme. We use the attribute data of all 52248 users with 6 attributes. In the second case, we randomly sample 1000 users from all users to conduct profile matching.

In both cases, we exam the similarity between all pairs of users as the ground truth. Then we run our profile matching protocols at different similarity levels. Figure 7 shows the number of candidate users of our protocol change with similarity requirement and the prime number p. The result shows that

TABLE 3
Comparison of efficiency with existing scheme.NC stands for non-candidate, and CD stands for candidate. $q = 256$

	Party	FNP [7]	FC10 [5]	Advanced [16]	Protocol 1
Computation	P_1 P_k	$(2m_t + m_k n)E_{2048}$ $m_k \log(\log m_t)E_{2048}$	$(2.5m_t n)M_{1024}$ $(m_t + m_k)E_{1024}$	$(3m_t n)E_{2048}$ $2m_t E_{2048}$	$(m_t+1)\mathcal{H} + m_t\mathcal{M} + \mathcal{E}$ $m_k\mathcal{H} + m_k\mathcal{M}$ (NC) $\kappa_k\gamma^2(\gamma+\beta) + (m_k+\kappa_k)\mathcal{H}$ $+m_k\mathcal{M} + \kappa_k\mathcal{D}$ (CD)
Communication bits	All	$8q(m_t + m_k n)$	$4qn(3m_t + m_k)$	$24[m_t m_k n$ $+tn(8m_t + 2m_k + 12m_t t)]$ $+16qm_t n$	$(1-\theta)32m_t^2 + (288 - q\theta)m_t$ $+q + qn * (\frac{1}{p})^{m_t\theta}$
Communication Transimission	All	1 broadcast n unicasts	$2n$ unicasts	$5n$ unicasts	1 broadcast $n * (\frac{1}{p})^{m_t\theta}$ unicasts

TABLE 7
Comparison of efficiency with existing scheme in typical scenario. NC stands for non-candidate, and CD stands for candidate. $m_t = m_k = 6$, $\gamma = \beta = 3$, $p = 11$, $n = 100$, $t = 4$.

	Party	FNP [7]	FC10 [5]	Advanced [16]	Protocol 1
Computation	P_1 P_k	$612E_{2048}$ $5E_{2048}$	$1500M_{1024}$ $12E_{1024}$	$1800E_{2048}$ $12E_{2048}$	$7\mathcal{H} + 6\mathcal{M} + \mathcal{E}$ (P_1) $6\mathcal{H} + 6\mathcal{M}$ (NC) $54\kappa_k + (6+\kappa_k)\mathcal{H} + 6\mathcal{M} + \kappa_k\mathcal{D}$ (CD)
Computation(ms) On phone	P_1 P_k	120564 985	225 408	354600 2364	0.7 (P_1) 0.63 (NC) $1.8\kappa_k + 0.63$ (CD)
Communication(KB)	All	151	300	704	0.22
Communication Transimission	All	1 broadcast 100 unicasts	200 unicasts	500 unicasts	1 broadcast $<$ 100 (#candidates unicasts)

TABLE 6
Decomposed computation time of our protocols.(ms)

Laptop			
	Mean	Min	Max
MatrixGen	7.2×10^{-3}	1.0×10^{-3}	2.4×10^{-2}
KeyGen	8.1×10^{-3}	2.3×10^{-3}	2.5×10^{-2}
RemainderGen	1.9×10^{-3}	3.1×10^{-4}	6.2×10^{-3}
HintGen	4.7×10^{-3}	2.8×10^{-4}	5.6×10^{-2}
HintSolve	3×10^{-2}	1.1×10^{-3}	1.1
Phone			
	Mean	Min	Max
MatrixGen	2.6×10^{-1}	4.4×10^{-2}	8.9×10^{-1}
KeyGen	6.3×10^{-2}	4.8×10^{-2}	1.4×10^{-1}
RemainderGen	3.4×10^{-1}	5.7×10^{-2}	1.14
HintGen	1.2	1.4×10^{-1}	12
HintSolve	6.9	2.6×10^{-1}	250

in both cases, the number of candidate users approaches the number of true matching users with increasing p. And a small p can achieve small size of candidate users and significantly reduce unwanted computation and communication cost for unmatching users.

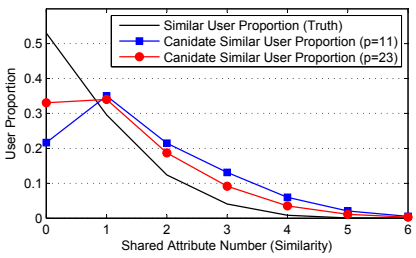
There is a worry that, the candidate profile key set may be very large for candidate users. Figure 8 presents the number of candidate profile keys during the matching with different similarity level and prime p. The result shows that in a real social networking system, the candidate key set is small enough to achieve efficient computation and communication for candidate users.

6 Related Work

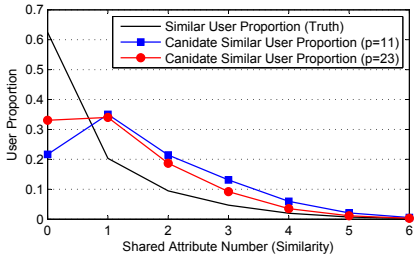
6.1 Privacy Preserving Friending

Most previous private matching work are based on the secure multi-party computation (SMC) [11]. There are two mainstreams of approaches to solve the private profile-based friending problem. The first category is based on private set intersection (PSI) and private cardinality of set intersection (PCSI) [16], [27]. Early work in this category mainly address the private set operation problem in database research, *e.g.* [5], [7], [15]. Later on some work treat a user's profile as multiple attributes chosen from a public set of attributes [16], [27] provide well-designed protocols to privately match users' profiles based on PSI and PCSI. The second category is based on private vector dot product [10]. [6], [34] considers a user's profile as vector and use it to measure social proximity. A trusted central server is required to precompute users social coordinates and generate certifications and keys. [35] improves these work with a fine-grained private matching by associating numerical values with attributes to indicate the interest levels. [13] enables the collusion-tolerable private computing without any secure channel. However, in the PSI based schemes, any user can learn the profile intersection with any other user. The PCSI and dot product based approaches cannot support a precise profile matchings. For example, they cannot help a user to find a match with attributes "doctor" and "female".

These protocols often rely on public-key cryptosystem and/or homomorphic encryption which results in expensive computation cost and usually requires a trusted third party. Even unmatched users involve in the expensive computation. Multiple rounds of interactions are required to perform the private profile matching between each pair of parties. They all need preset procedure, *e.g.* exchanging public keys before matching, precomputing vector [6], establishing secure communication channel and share secret [16]. Furthermore, these protocols are unverifiable. These imitations hinder the SMC-related protocols to get practical in decentralized social networks.

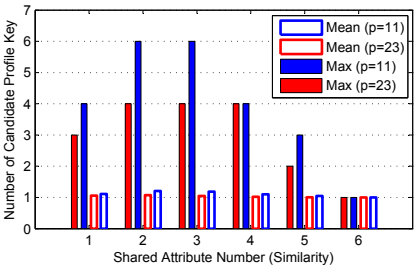


(a) users with 6 attributes

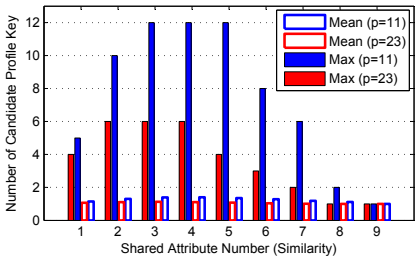


(b) diverse numb. of attributes

Fig. 7. Candidate user proportion with different similarity and prime number.



(a) users with 6 attributes



(b) diverse num. of attributes

Fig. 8. Size of candidate profile key set with different similarities.

6.2 Establishing Secure Channel

Secure communication channel construction is very important in practical private friending system but is often ignored. Secure communication channels are usually set up by authenticated key agreement protocols. This can be performed by relying on a public-key infrastructure, e.g., based on RSA or the Diffie-Hellman protocol.The public-key based methods allow parties to share authenticated information about each other, however they need a trusted third party. Although Diffie-Hellman key exchange method allows two parties to jointly establish a shared secret key, it is known to be vulnerable to the Man-in-the-Middle attack.

Device pairing is a another technique to generate a common secret between two devices that shared no prior secrets with minimum or without additional hardware, *e.g.*, [25] and [21]. However, they employ some out-of-band secure channels to exchange authenticated information and the interaction cost is not well suited to MSN where secure connections are needed immediately between any users. With these existing schemes, it is more complicated to establish a group key.

6.3 Attribute Based Encryption

Attribute based encryption is designed for access control of shared encrypted data stored in a server, which was introduced by Sahai and Waters [22]. Only the user possessing a certain set of credentials or attributes is able to access data. Then ABE was improved [3], [4], [8]. [8] develops the Key-Policy Attribute-Based Encryption (KP-ABE) for fine-grained sharing of encrypted data. [3] presents a Ciphertext-Policy Attribute-Based Encryption (CP-ABE) which keeps encrypted data confidential even if the storage server is untrusted. [12] proposes a method providing anonymity for ABE. [4] gives a construction for a multi-authority attribute-based encryption system, and [14] supports privacy-preserving multi-authority data access. All the ABE schemes rely on asymmetric-key cryptosystem, which cost expensive computation. And they require a complicate setup and a server. In this work, we design a symmetric-encryption based privacy-preserving profile matching mechanism, which is significantly more efficient and requires no setup or server.

7 Conclusions

In this paper, we design a novel symmetric key encryption based privacy-preserving profile matching and secure communication channel establishment mechanism in decentralized MSN without any presetting or trusted third party. Several protocols were proposed for achieving verifiability and different levels of privacy. We analyzed the performance of our protocols and compared them with existing protocols. We conducted extensive evaluations on the performances using a large scale dataset from real social networking. The results show that our mechanisms outperform existing methods significantly and provide efficient and secure solution for mobile social networks. Our efficient techniques, including private fuzzy attribute matching and secure communication channel establishing, can also be applied to many other scenarios where parties don't necessarily trust each other, *e.g.*, advertising auction, information sharing and location based services. In our future work, we will integrate these techniques into more networking systems.

Acknowledgment

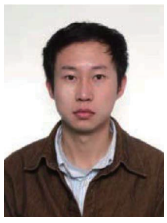
This research is partially supported by NSFC 61125020, NSFC CERG-61361166009, NSFC 61103187, NSFC 61472211, NSFC Major Program 61190110. The research of Li is partially supported by NSF CNS-1035894, NSF ECCS-1247944, NSF ECCS-1343306, NSF CMMI 1436786, NSFC 61170216, NSFC 61228202.

REFERENCES

[1] "Magnetu," http://magnetu.com.
[2] "Tencent weibo," http://t.qq.com/.
[3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of IEEE S&P*, 2007.
[4] M. Chase, "Multi-authority attribute based encryption," *Theory of Cryptography*, pp. 515–534, 2007.
[5] E. De Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *Financial Cryptography and Data Security*, 2010, pp. 143–159.
[6] W. Dong, V. Dave, L. Qiu, and Y. Zhang, "Secure friend discovery in mobile social networks," in *Proceedings of IEEE INFOCOM*, 2011.
[7] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *EUROCRYPT*, 2004, pp. 1–19.
[8] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of ACM CCS*, 2006.
[9] B. Han and T. Baldwin, "Lexical normalisation of short text messages: Makn sens a# twitter," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011, pp. 368–378.
[10] I. Ioannidis, A. Grama, and M. Atallah, "A secure protocol for computing dot-products in clustered and distributed environments," in *Proceedings of IEEE ICPP*, 2002.
[11] T. Jung, X. Mao, X.-Y. Li, S. Tang, W. Gong, and L. Zhang, "Privacy-preserving data aggregation without secure channel: multivariate polynomial evaluation," in *Proceedings of IEEE INFOCOM*, 2013.
[12] T. Jung, X.-Y. Li, Z. Wan, and M. Wan, "Control cloud data access privilege and anonymity with fully anonymous attribute based encryption," in *Transactions on Information Forensics and Security*. IEEE, 2014.
[13] T. Jung and X.-Y. Li, "Collusion-tolerable privacy-preserving sum and product calculation without secure channel," in *Transactions on Dependable and Secure Computation*. IEEE, 2014.
[14] T. Jung, X.-Y. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in *Proceedings of INFOCOM*. IEEE, 2013.
[15] L. Kissner and D. Song, "Privacy-preserving set operations," in *Advances in Cryptology–CRYPTO*, 2005, pp. 241–257.
[16] M. Li, N. Cao, S. Yu, and W. Lou, "Findu: Privacy-preserving personal profile matching in mobile social networks," in *Proceedings of IEEE INFOCOM*, 2011.
[17] X.-Y. Li and T. Jung, "Search me if you can: privacy-preserving location query service," in *Proceedings of IEEE INFOCOM*, 2013.
[18] C. Y. Ma, D. K. Yau, N. K. Yip, and N. S. Rao, "Privacy vulnerability of published anonymous mobility traces," *IEEE/ACM Transaction on Networking*, 2012.
[19] A. Mikheev, "Document centered approach to text normalization," in *SIGIR*. ACM, 2000, pp. 136–143.
[20] K. Okamoto, W. Chen, and X.-Y. Li, "Ranking of closeness centrality for large-scale social networks," in *FAW*, 2008.
[21] S. Pasini and S. Vaudenay, "Sas-based authenticated key agreement," in *PKC*, 2006.
[22] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *EUROCRYPT*, 2005, pp. 457–473.
[23] E. Sarigol, O. Riva, P. Stuedi, and G. Alonso, "Enabling social networking in ad hoc networks of mobile phones," in *Proceedings of VLDB Endowment*, 2009.
[24] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards, "Normalization of non-standard words," *Computer Speech & Language*, vol. 15, no. 3, pp. 287–333, 2001.
[25] F. Stajano and R. Anderson, "The resurrecting duckling: security issues for ubiquitous computing," *IEEE Computer*, vol. 35, no. 4, pp. 22–26, 2002.
[26] S.-J. Tang, J. Yuan, X. Mao, X.-Y. Li, W. Chen, and G. Dai, "Relationship classification in large scale online social networks and its impact on information," in *Proceedings of IEEE INFOCOM*, 2011.
[27] M. Von Arb, M. Bader, M. Kuhn, and R. Wattenhofer, "Veneta: Serverless friend-of-friend detection in mobile social networking," in *Proceedings of IEEE WIMOB*, 2008.
[28] Z. Yang, B. Zhang, J. Dai, A. C. Champion, D. Xuan, and D. Li, "E-smalltalker: A distributed mobile system for social networking in physical proximity," in *Proceedings of IEEE ICDCS*, 2010.
[29] C. Zhang, J. Sun, X. Zhu, and Y. Fang, "Privacy and security for online social networks: challenges and opportunities," *IEEE Network*, vol. 24, no. 4, pp. 13–18, 2010.
[30] L. Zhang, X.-Y. Li, Y. Liu, Q. Huang, and S. Tang, "Mechanism design for finding experts using locally constructed social referral web," in *TPDS*, 2013.
[31] L. Zhang, X. Ding, Z. Wan, M. Gu, and X. Li, "Wiface: A secure geosocial networking system using wifi-based multi-hop manet," in *ACM MobiSys MCS workshop*, 2010.
[32] L. Zhang, T. Jung, P. Feng, X.-Y. Li, and Y. Liu, "Cloud-based privacy preserving image storage, sharing and search," *arXiv preprint arXiv:1410.6593*, 2014.
[33] L. Zhang, T. Jung, C. Liu, X. Ding, X.-Y. Li, and Y. Liu, "Outsource photo sharing and searching for mobile devices with privacy protection," *arXiv preprint arXiv:1410.6589*, 2014.
[34] L. Zhang, X.-Y. Li, Y. Liu, and T. Jung, "Verifiable private multi-party computation: Ranging and ranking," in *Proceedings of IEEE INFOCOM*, 2013.
[35] R. Zhang, Y. Zhang, J. Sun, and G. Yan, "Fine-grained private matching for proximity-based mobile social networking," in *Proceedings of IEEE INFOCOM*, 2012.
[36] Y. Zheng, M. Li, W. Lou, and Y. T. Hou, "Sharp: Private proximity test and secure handshake with cheat-proof location tags," in *ESORICS*, 2012.

Lan Zhang received her Bachelor degree (2007) in School of Software at Tsinghua University, China, and her Ph.D. degree (2014) in the department of Computer Science and Technology, Tsinghua University, China. She is now a Post Doctor in the School of Software, Tsinghua University, China. Her research interests span social networks, privacy, secure multi-party computation and mobile computing, etc.

Xiang-Yang Li is a professor at the Illinois Institute of Technology. He holds EMC-Endowed Visiting Chair Professorship at Tsinghua University. He currently is a distinguished visiting professor at XiAn JiaoTong University, and University of Science and Technology of China. He is a recipient of China NSF Outstanding Overseas Young Researcher (B). Dr. Li received MS (2000) and PhD (2001) degree at Department of Computer Science from University of Illinois at Urbana-Champaign, a Bachelor degree at Department of Computer Science and a Bachelor degree at Department of Business Management from Tsinghua University, P.R. China, both in 1995. His research interests include wireless networking, mobile computing, security and privacy, cyber physical systems, smart grid, social networking, and algorithms.



Kebin Liu received his BS degree in Department of Computer Science from Tongji University in 2004, and MS and Ph.D. degrees in Shanghai Jiaotong University, in 2007 and 2010. He is currently an assistant researcher in the School of Software and TNLIST, Tsinghua University. His research interests include WSNs and distributed systems.

Taeho Jung received the B.E degree in Computer Software from Tsinghua University, Beijing, in 2007, and he is working toward the Ph.D degree in Computer Science at Illinois Institute of Technology while supervised by Dr. Xiang-Yang Li. His research area, in general, includes privacy & security issues in mobile network and social network analysis. Specifically, he is currently working on the privacy-preserving computation in various applications and scenarios.

Yunhao Liu received his BS degree in Automation Department from Tsinghua University, China, in 1995, and an MS and a Ph.D. degree in Computer Science and Engineering at Michigan State University in 2003 and 2004, respectively. He is now Chang Jiang Professor and Dean of School of Software, Tsinghua University, China.