PriMatch: Fairness-aware Secure Friend Discovery Protocol in Mobile Social Network

Muyuan Li, Zhaoyu Gao, Suguo Du, Haojin Zhu
Shanghai Jiao Tong University, Shanghai, China
{leilmyxwz, oversky710, sgdu, zhu-hj}@sjtu.edu.cn,

Mianxiong Dong, Kaoru Ota
The University of Aizu
{mx.dong, k.ota}@ieee.org

***Abstract*—Mobile social networks are expected to substantially enrich interaction with ubiquitous computing environments by integrating social context information into local interactions. However, in mobile social networks, the mobile users may face the risk of leaking their personal information and their location privacy. In this study, we first model the secure friend discovery process as a generalized privacy-preserving interest/profile matching problem. Then, we identify a new security threat arising from existing secure friend discovery protocols, coined as *runaway attack*, which is expected to introduce serious fairness issue. To address this new threat, we introduce a novel blind vector transformation technique, which could hide the correlation between the original vector and the transformed result. Based on it, we propose our fairness-aware privacy preserving interest/profile matching protocol, which enables one party to match its interest with the profile of another, without revealing its real interest and profile and vice versa. The detailed security analysis as well as real-world implementations demonstrate the effectiveness and the efficiency of the proposed protocol.**

***Index Terms*—Mobile Social Network, Friend Discovery, Profile/Interest Match, Privacy Preserving, Fairness**

I. Introduction

Mobile social networks are expected to substantially enrich interaction with ubiquitous computing environments by integrating social context information into local interactions. Location-aware mobile social networks such as Foursquare[1] and Gowalla[2] provide the opportunity to leverage social networking context within a local physical proximity using mobile smart phones. As a valuable complement to web-based online social networking, mobile social networks allow mobile users to have more tangible face-to-face social interactions in public places such as bars, airports, trains, and stadiums [3]. Profile matching is more than important for fostering the wide use of mobile social networks because finding the nearby individuals of similar interests is always the first step for any social networking.

However, the existing mobile social network systems pay little heed to the security and privacy concerns associated with revealing one's personal social networking preferences and friendship information to the ubiquitous computing environment. In particular, in mobile social networks, the mobile users may face the risk of leaking their personal information and their location privacy. Under this circumstance, the attackers can directly associate the personal profiles with real persons nearby and then launch more advanced attacks.

Recently, there are quite a few proposals for *Private Profile Matching*, which allow two users to compare their personal profiles without revealing private information to each other [4], [5]. In a typical private profile matching scheme, the personal profile of a user consists of multiple attributes chosen from a public set of attributes (e.g., various interests[5], disease symptoms[6], or friends[7] etc.). The private profile matching problem could then be converted into Private Set Intersection (PSI) [8], [9] or Private Set Intersection Cardinality (PSI-CA) [10], [11]. In particular, two mobile users, each of whom holds a private data set respectively, could jointly compute the intersection or the intersection cardinality of the two sets without leaking any additional information to either side.

However, there are quite a few challenges which make the existing private profile matching solutions less practical in applications. For example, similar to most of the online social network applications, a mobile social networking user is expected to freely search its potential common-interest friends by matching his *interest* with the personal *profiles* of the searching targets rather than making the profile matching directly. This could be well supported by the dating social networks, in which a member may seek another member satisfying some particular requirements (e.g., gender, age ranges or even living location as in [12]). Further, the existing proposals are one-way only and profile matching requires running a protocol twice, with reversed roles in the second run. This two-pass protocol may be exploited by the dishonest user or even a malicious attacker to launch the *runaway attack*, in which one that wants to learn another user's interests but is unwilling to reveal his own interests can simply abort the protocol in the second round. This runaway attack incurs a serious fairness issue. The runaway attack may be more challenging in the case of separating user's profile from his interest since matching the users' profiles and interests could only be achieved in the two steps.

To solve the above mentioned challenges and thus further enhance the usability of mobile social networks, we present PriMatch, a novel fairness-aware secure friend discovery protocol. In PriMatch, a successful matching only happens in case that the interests of both of the participants could match the profiles of the others. In other words, no one can learn any extra information from the protocol unless another participant is exactly what he is looking for and vice versa. PriMatch is motivated from a simple observation that if two vectors match,

978-1-4673-0921-9/12/$31.00 ©2012 IEEE

they will still match no matter whether they are transformed in the same way (e.g., add or substract a randomly generated vector) or shuffled with the same order.

To achieve this goal, we introduce a novel *Blind Vector Transformation* technique, under which each participant contributes a part of the vector transformation while any single one of the parties cannot recover the original vectors from the final transformation result. Therefore, with blind vector transformation, we could enable a party to match its interests with another's profile but, at the same time, to keep the interests/profiles private. Further, to thwart runaway attack, we introduce a lightweight verifier checking technique, which enables the verifier to check the matching at the minimized overhead and prevent any participant from launching the runaway attack.

The contribution of this work could be summarized as follows:

- For the first time, we separate the user's interest from his profile, which is expected to be a generalization of traditional profile matching problem.
- We introduce a novel blind vector transformation technique, which could hide the correlation between the original vector and the transformed result. Based on it, we propose our PriMatch protocol, which enables one party to match its interest with the profile of another, and vice versa, without revealing their real profile and interest.
- We introduce a novel lightweight verifier checking approach to thwart runaway attack and thus achieve the fairness of the two participants.
- We implement our protocol in real experiments. We demonstrate the performance of the proposed scheme via extensive experiment results.

The remaining of this paper is organized as follows: Section II defines our system model which involves a novel private match interaction and then presents the threats and our design goals in this paper. We propose a fairness-aware secure friend discovery protocol – PriMatch in Section III. Section IV proves the security of PriMatch by probability analysis. We show the efficiency of PriMatch by real implementation in Section V and we conclude this paper in Section VI.

II. Problem Formulation and Preliminary

In this section, we first introduce our system model as well as the adversary model, then we present our definition for private fair friend discovery.

A. System Model

In mobile social networks, a user may launch a query process to find potential friends. Each user initializes a profile as his inherent characteristic. The profile consists of multiple attributes (e.g., user's occupation, hobbies and other private information) which could be denoted as a vector $P = \{p_1, p_2, \ldots, p_n\}$. Here, $p_j (j = 1, \ldots, n)$ refers to one attribute of P and is an integer. When a user U issues a query, he firstly generates the corresponding interest vector $I = \{i_1, i_2, \ldots, i_n\}$. Note that, similar to a typical search process in online social networks, the user could freely generate different interests before each query.

A typical friend discovery process could be described as follows. A user A sends his current interest I_A to user B, and then he gets B's current interest I_B. After the interest exchange, A compares his own profile P_A with I_B while B compares his profile P_B with I_A. We define a successful matching as P_A matches I_B and, at the same time, P_B matches I_A, which is similar to the privacy level introduced in [5].

Note that, in some cases, the user may only care about some specific attributes. To handle those attributes that are not concerned in one query, we classify the attributes of I into interested attributes (IA) and non-interested attributes (NIA). Therefore, a successful matching should ensure the IA fields of the interests and profiles match while NIA fields mismatch. In this work, to make NIA fields not affect the comparison result, we could simply assign an impossible value to the NIA fields of the interest, which directly makes the comparison fail no matter what the value of the profile is. For example, we could fill the NIA fields with values out of the valid range.

We also assume the existence of third-party verifiers. These verifiers are either honest, semi-honest or actively malicious. The minority of them may collude with either user involved.

B. Adversary Model

we consider an adversary who is curious about others' profile and interest. Therefore, without an appropriate security countermeasure, the friend discovery process may suffer from a series of privacy threats. In particular, we consider the following adversary model

1) **Interest/profile Leaking Attack:** The adversary may try to infer the interests or the profiles of other users during the interest/profile matching process.
2) **Runaway Attack (Protocol Abortion Attack):** Under this attack, even with a privacy preserving interest/profile matching protocol, the adversary can obtain the personal information of another user by aborting the protocol anytime and perform certain analysis over the information already obtained.
3) **Collusion Attack:** The adversary may collude with other users to try to infer the user's private information.

Therefore, we define the *Private Fair Friend Discovery* that should satisfy the following objectives:

1) **Privacy Guarantee:** The protocol should well protect the personal information of the mobile users. In particular, no one including the external and internal attackers could get the profile/interest information of the users. In the proposed protocol, after performing the privacy-preserving file/interest matching protocol, each participant could only obtain the comparison result "success" or "fail". No other information will be leaked from the protocol.
2) **Fairness Assurance:** In each phase of the protocol, a user can obtain personal information from others as

much as his own personal information leaking from the protocol. In other words, no one can gain more information than what he's told others, which ensures the fairness of the protocol.

C. Fast Variant of Paillier Homomorphic Encryption

The protocol proposed in this paper involves Paillier's homomorphic encryption. In the fast variant of Paillier cryptosystem [13], given a message m and the public key (n, g), the encryption function can be described as $E(m) = g^{(m+rn)}(mod\ n^2)$, where r is a random factor in the Paillier cryptosystem. The decryption function is $D_k(\alpha) = \frac{L(c^\alpha mod\ n^2)}{L(g^\alpha mod\ n^2)}$, where $L(x) = \frac{x-1}{n}$ and $\alpha \leq n$. It satisfies the following homomorphic property:

$$E(m_1)\cdot E(m_2) = g^{(m_1+m_2)+n(r_1+r_2)}(mod\, n^2) = E(m_1+m_2)$$

We denote the encryption of profile $P = (p_1, p_2, \ldots, p_n)$ under public key k as $E_k[P] = (E_k(p_1), E_k(p_2), \ldots, E_k(p_n))$ while $D_k[P]$ is the decryption function, the same with interest I. We assume that each user i has his own Paillier public and private key (pk_i, sk_i), and the encryption and decryption in this paper are all executed in Paillier cryptosystem.

III. Private Fair Match Protocol

A. Protocol Overview

The basic idea of blind vector matching is allowing two untrusted parties to transform two vectors into *blind* ones by following a series of private and identical steps, e.g., adding a random vector, shuffling in the same order. Since the transformation follows the same step, the matching results (e.g. the number of matches in the interest/profile) keep unchanged before and after the transformation, which enables the untrusted participants to compare the interest/profile without leaking their real interest/profile information.

The major challenge of the blind vector is how to hide the real value of the interest/profile of the participants. The basic idea is that two untrusted participants contribute a part of this transformation and each of them cannot recover the real interest/profile. To enable that, we define five operation primitives as follows:

execution	function	operation
Encrypt	$Encrypt(\vec{v}, pk)$	Encrypt the vector $\vec{v}$ by Paillier Encryption with public key pk as $E_{pk}[\vec{v}]$
VecAdd	$VecAdd(\vec{v}, \vec{r})$	Add a random number vector $\vec{r}$ to $\vec{v}$. If both vectors are encrypted under Paillier Encryption, the adddition is performed as $E[\vec{v}]E[\vec{r}] = E[\vec{v} + \vec{r}]$
VecExt	$VecExt(\vec{v}, \vec{r})$	Append some dummy vector $\vec{r}$ to $\vec{v}$ to obtain $v\|\|r$.
VecShuffle	$VecShuffle(\vec{v})$	Randomly shuffle the elements in vector $\vec{v}$
VecRev	$VecRev(\vec{v}, k)$	Randomly change the value of k elements in vector $\vec{v}$

TABLE I
The execution in PriMatch

The overall procedure could be described as follows: the user A encrypts his profile with his own public key by triggering operation $Encrypt(\vec{v}, pk)$. Here, Paillier is adopted since it keeps A's profile private and, at the same time, enables B to perform blind transformation on it. The transformation operations include $VecAdd$, $VecExt$, $VecShuffle$, $VecRev$, which are summarized in Table I. To ensure the fairness of the protocol, we require that each participant should compare the blinded interest and profile. Each participant sends the number of matching vector pairs as well as the size of search interest to some verifiers. The verifiers compare if the number of search interest equals the number of matching vector pairs. If both of two sides match, the verifiers informs two sides of a successful match. In this process, any participant learns nothing about the personal information of the other except match or not. In the follows, we present the detailed protocol.

B. The Detailed Protocols

1) System Initialization: U_1 and U_2 generates their Paillier's private key and public key pairs denoted as (sk_1, pk_1) and (sk_2, pk_2).

Algorithm 1: The Blind Transformation Algorithm

Input: $\dot{P}_a \leftarrow U_a$'s profile encrypted under his public key pk_a, $I_b \leftarrow U_b$'s interest, $e_b \leftarrow$ the number of interest U_b considers in I_b and $l_b \leftarrow$ a security parameter.

Output: $\ddot{P}_a \leftarrow$ the blind-transformed profile vector for U_a, $\ddot{I}_b \leftarrow$ the transformed interest vector for U_b and $s_b \leftarrow$ the actual matching result for U_b.

function BLIND-TRANSFORMATION($\dot{P}_a$, pk_a, I_b, e_b, l_b)
- $r_b \leftarrow$ random vector of length $n = ||\dot{P}_a||$
- $\dot{r_b} \leftarrow Encrypt(r_b, pk_a)$
- $\widetilde{P_a} \leftarrow VecAdd(\dot{P}_a, \dot{r_b})$
- $\widetilde{I_b} \leftarrow VecAdd(I_b, r_b)$
- $y_b \leftarrow$ random vector of length l_b
- $\dot{y_b} \leftarrow Encrypt(y_b, pk_a)$
- $\dot{P}_a \leftarrow VecExt(\widetilde{P_a}, \dot{y_b})$
- $k_b \leftarrow$ random number in $[1, l_b]$
- $\widetilde{y_b} \leftarrow VecRev(y_b, k_b)$
- $\dot{I}_b \leftarrow VecExt(\widetilde{I_b}, \widetilde{y_b})$
- $\ddot{I}_b \leftarrow VecShuffle(\dot{I}_b)$, $\ddot{P}_a \leftarrow VecShuffle(\widetilde{P_a})$ } with the same order
- $s_b \leftarrow e_b + l_b - k_b$
- **return** $\ddot{P}_a, \ddot{I}_b, s_b$

end function

2) The Blind Transformation Phase: We consider two users U_1 and U_2 are performing a friend discovery process and their profiles are denoted as P_1 and P_2. For this matching, U_1 and U_2 may only consider e_1 and e_2 out of total n interest fields. Thus, there are $n - e_1$ and $n - e_2$ attributes out of the match scope for U_1 and U_2, respectively. We assume the current interest vectors are I_1 and I_2.

In the blind transformation phase, each participant will encrypt his profile by using his public key and provide it to the other side for blind transformation. In the follows, we introduce the blind transformation process by taking U_2 transforming U_1's profile and his own interest as an example. It is similar for U_1 to blind transform U_2's profile. U_1 performs $Encrypt(P_1, pk_1)$ to encrypt his profile P_1, which is denoted as $\dot{P}_1$. U_1 sends $\dot{P}_1$ and pk_1 to U_2. Then, U_2 performs the following blind transformation operation:

- **Blind Add:** U_2 generates a random vector r_2, and then performs $\dot{r_2} = Encrypt(r_2, pk_1)$. After that, U_2 calculates $\widetilde{P_1} = VecAdd(\dot{P}_1, \dot{r_2})$ and $\widetilde{I_2} = VecAdd(I_2, r_2)$ by adding $\dot{r_2}$ to $\dot{P}_1$ and r_2 to I_2, respectively.
- **Blind Append:** U_2 generates a random vector y_2 of length l_2, where l_2 is a predetermined security parameter, and then performs $\dot{y_2} = Encrypt(y_2, pk_1)$ to get $\widetilde{\dot{P}_1} = VecExt(\widetilde{P_1}, \dot{y_2})$.
- **Blind Reverse:** U_2 randomly selects $k_2 \in \{1, 2, \cdots, l_2\}$ and performs $\widetilde{y_2} = VecRev(y_2, k_2)$, then obtains $\dot{I}_2 = VecExt(\widetilde{I_2}, \widetilde{y_2})$.
- **Blind Shuffle:** U_2 performs $\ddot{I}_2 = VecShuffle(\dot{I}_2)$ and $\ddot{P}_1 = VecShuffle(\widetilde{\dot{P}_1})$ with the same order.

After performing this process, U_2 finishes the blind transformation of P_1 and I_2. In the same time, U_2 also encrypts his profile and U_1 follows a similar strategy to make a blind transformation towards P_2 and I_1.

Note that, among the above four operations, $VecAdd$ and $VecShuffle$ are used to conceal the original value of P_1 and prevent U_1 from obtaining the transformation ways of U_2 by linking P_1 and $\ddot{P}_1$. U_1 (or U_2) can still obtain the correct number of matched interest/profiles since P_1 and I_2 (or P_2 and I_1) follow the same transformation pattern. However, U_1 or U_2 can choose to launch runaway attack by obtaining the matching number of his interest and another participant's profile without revealing his own profile. In the proposed fair protocol, we further introduce $VecExt$ and $VecRev$, which are used to hide the actual matching numbers. In particular, on U_1 side, $VecExt$ introduces extra l_2 matched attributes to original matching result while $VecRev$ introduces k_2 mismatched ones. Therefore, the actual matching result is updated to $s_2 = e_2 + l_2 - k_2$ for U_2 and $s_1 = e_1 + l_1 - k_1$ for U_1. The blind transformation phase is summerised in Algorithm 1.

3) Fair Matching Phase: After decryption, U_1 compares $\hat{P}_1 = Decrypt(\ddot{P}_1, sk_1)$ with $\ddot{I}_2$ to get the number of matched entries $\hat{s_2}$, while U_2 could get $\hat{s_1}$ similarly. U_1 sends $h_1 = H(s_1||\hat{s_2})$(H is a crypto hash function) whereas U_2 sends $h_2 = H(\hat{s_1}||s_2)$ to some verifiers, who are selected randomly by U_1 and U_2, to verify whether $h_1 = h_2$. If $h_1 = h_2$, the match succeeds, otherwise, it fails.

4) Discussions: The proposed basic protocol cannot tolerate collusion attack. Suppose U_1 and a verifier V collude, when V receives U_2's comparing result $h_2 = H(\hat{s_1}||s_2)$, he sends it to U_1. Since $\hat{s_1} \leq (n + l_1)$, $s_2 \leq (n + l_2)$ and n, l_1 and l_2 are all limited due to the efficiency of encryption and decryption, U_1 could get $\hat{s_1}$ by brute force in worst-case time complexity of $O((n + l_1)(n + l_2))$. If U_1 could get $\hat{s_1}$, based on the information he obtains, he could find out how many attribute matches are in P_2 and I_1. Then he could figure out U_2's privacy by a probability that can't be neglected.

C. Tolerate Collusion Attack via Blind Linear Transformation

In this section, we propose an advanced scheme Blind Linear Transformation to tolerate the collusion attack(similar technique has been used in [14]). The basic idea is that, instead of directly sending h_1 and h_2 to the verifiers, an additional blind linear transformation round is introduced to protect the hash result. The scheme is shown as follows.

1) U_1 concatenates s_1 and $\hat{s_2}$ to get a number $sr_1 = s_1||\hat{s_2}$. He then sends $E_{pk_1}[sr_1]$ to U_2.
2) U_2 generates a pair of random numbers (a_2, k_2). He then computes $E_{pk_1}[s\dot{r}_1] = E_{pk_1}[a_2 sr_1 + k_2]$. He also gets $sr_2 = \hat{s_1}||s_2$ and transforms sr_2 in the same way to obtain $s\dot{r}_2 = a_2 sr_2 + k_2$.
3) U_2 sends $E_{pk_2}[s\dot{r}_2]$ and $E_{pk_1}[s\dot{r}_1]$ back to U_1.
4) As in 2), U_1 selects a pair of random numbers (a_1, k_1) and computes $E_{pk_2}[s\ddot{r}_2] = E_{pk_2}[a_1 s\dot{r}_2 + k_1]$. Decrypting with sk_1, he obtains $s\dot{r}_1$. He gets $s\ddot{r}_1 = a_1 s\dot{r}_1 + k_1$ and then sends $E_{pk_2}[s\ddot{r}_2]$ back to U_2.
5) U_2 decrypts $E_{pk_2}[s\ddot{r}_2]$. U_1 sends $H(s\ddot{r}_1)$ and U_2 sends $H(s\ddot{r}_2)$ to verifiers to test their equality.

In this way, each side hides a bijection $f : x \to y$ from the other, providing security assurance for sr_1 and sr_2.

IV. Security Analysis

In this section, we will demonstrate the fairness and the privacy of PriMatch by the detailed security analysis.

A. Security Against Interest/Profile Leaking

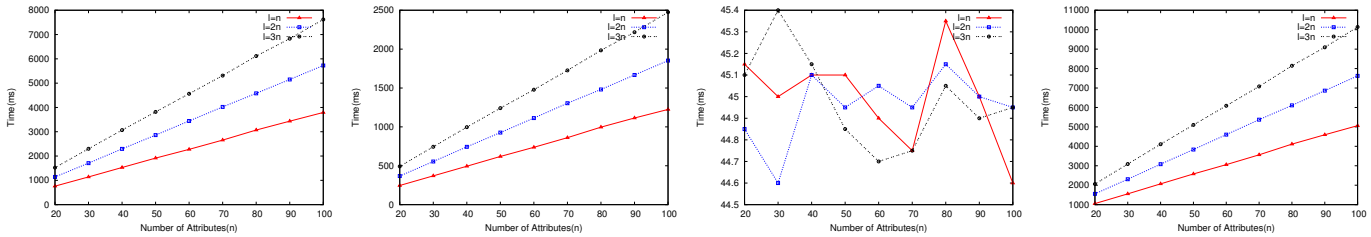
Without loss of generality, we just consider P_1 and I_2. Since the profile P_1 is encrypted by Pallier cryptosystem, and without the secret key sk_1, no one except U_1 could get P_1. Thus the privacy in P is preserved.

The privacy of I_2 is guaranteed by PriMatch. After receiving the processed P_1 and I_2, U_1 can not correlate any item of I_2 with the attributes in P_1. At the same time, it is guaranteed for U_2 that U_1 can not test his interest by changing P_1 arbitrarily.

B. Security Against Runaway Attack

As we have introduced in Section III, after U_1 decrypts the processed P_1, he could obtain the comparison result $\hat{s_2}$ which indicates how many pairs of items between processed P_1 and I_2 match. If he knows $m_2 = l_2 - k_2$ which indicates how many pairs are the same in appended vector, he will know that there are $s_2 - m_2$ pairs of matched attributes between P_1 and I_2. Therefore, U_1 could randomly select $s_2 - m_2$ attributes in his profile P_1 as the corresponding attributes in I_2. If this probability could not be neglected, U_1 may abort the protocol and get some of U_2's interest with a high probability.

We will use the following theorem to discuss the upper bound of the successful probability that U_1 could guess any item of I_2 without any error.



(a) Blind Transformation (b) Fair Matching (c) Blind Linear Transformation (d) Total time elapsed

Fig. 1. Run time in different phases of PriMatch

Theorem 1 Given a profile P and an interest I which have been processed by PriMatch, the correct-guess probability $P(CG)$ that U could infer any item of I based on the processed P and the comparing result s is bounded by $\frac{3}{ln}(n \geq 5)$, where n is the length of P and l is the number of attributes appended to P.

Proof: The successful guess probability is expressed as:

$$P(CG) = \sum_{m'=1}^{\min(s,l)} p(m = m')Pr\{CG|s, m\} \quad (1)$$

where $p(m = m')$ is the probability that U_1 could guess m correctly, and in our scheme, $p(m = m') = \frac{1}{l}, m' \in \{1, 2, \ldots, l\}$. $Pr\{CG|s, m\}$ is the probability that given s and m, U_1 could guess $s - m$ items correctly. Obviously, when $s - m \geq n$, $P(CG|s, m) = 0$, when $s - m < n$, $P(CG|s, m) = \frac{1}{\binom{n}{s-m}}$. No matter $s > l$ or $s \leq l$, we get

$$P(CG) = \frac{1}{l} \sum_{m=1}^{s-1} \frac{1}{\binom{n}{s-m}} \quad (2)$$

By mathematical induction, $P(CG) \leq \frac{3}{ln}(n \geq 5)$. ■

Theorem 1 indicates that given ϵ as the expected secure probability such that $P(CG) < \epsilon$, if ϵ is small enough, then we think U_1 will get nothing about U_2's interest since he could not guess any part of I_2 correctly, thus he has no incentive to abort the protocol. Furthermore, we could safely bound it with $\frac{3}{ln} < \epsilon$. And according to this inequality, we can calculate l and m to guarantee the fairness. Theorem 1 also indicates that if two users are not matched finally, they can't guess anything according to the comparing result. Thus the fairness is guaranteed by PriMatch.

C. Security Against Collusion Attack

In the revealing phase, the number of matches on both sides will be transformed and neither side knows how the other side performs such transformation. Obviously, the probability of guessing $(a_i, b_i), i \in (1, 2)$ of the other side is negligible. We have the following theorem.

Theorem 2 Given the knowledge of sr_1 and $s\dot{r}_1$ in $s\dot{r}_1 = a_2 sr_1 + k_2$, the probability that U_1 guesses (a_2, k_2) correctly $(sr_1, s\dot{r}_1 \in \mathbb{Z}_p^*)$ is bounded by $1/p$.

Proof: The attempt to guess the parameters can be formalized as guessing (a, b) in $y = ax + b$ given the knowledge of only one pair of (x_0, y_0). This problem is at least as hard as guessing y_1 in $y = ax + b$ with (x_0, y_0) and x_1, since if the previous problem can be solved, the adversary can use (x_0, y_0) to guess (a, b) and then calculate $y_1 = ax_1 + b$. The security of the latter problem is proved in [14] and the probability of guessing y_1 correctly is $1/p$. Thus, the probability of guessing (a_2, k_2) correctly is at most $1/p$. ■

Theorem 2 shows the security of our scheme against collusion attack. A user is unable to search over $[1, n+l]$ to obtain s and $\hat{s}$ without knowing $f : x \to y$. Even if he obtains $\dot{sr}$ correctly(by brute force trial and error with the hash value), he can only get $sr = (\dot{sr} - b)/a$ with negligible probability.

The Fairness Assurance in revelation is achieved in that users can choose to trust the majority of the verifiers and the only results revealed so far are "success" or "fail", which is known to both sides at the same time.

The verifiers only receive two hash values and should only answer whether they are equal. The only information available to them is the two hash values and the only answer they can get is whether they are equal, which is just as intended. No further information can be derived from the hash value due to one-way characteristic. Thus, the revealing phase reveals information no more than "success" or "fail".

V. Evaluation

We implement our protocol in Java and evaluate it on a laptop with Intel Core i3-330m(2.1GHz) and 2GB RAM. The Paillier encryption library is based upon [15]. We modify it to implement the fast variant of Paillier scheme as proposed in [13]. We evaluate the running time of our protocol in Blind Transformation, Fair Matching and Blind Linear Transformation phase. In Blind Shuffle phase, we use Knuth Shuffle[16] in order to guarantee the randomness in permutation.

The Paillier key length is selected as 1024-bit. The α is 160-bit as in [13]. We tested a single round in our simulation in which U_1 launches the protocol and matches his profile against U_2's interest. As in a real world application such protocol will be executed in parallel by two users, a single round is enough to measure its efficiency. The results are depicted in Figure 1. We choose 3 security parameter length $l = n$, $l = 2n$, $l = 3n$. The number of attributes ranges from 20 to 100. We measure the running time against the number of attributes under those 3 parameter settings. We've plotted the average value of 20 runs. Detailed statistics about the results is shown in Table II

n	l	Blind Transformation(ms)			Fair Matching(ms)			Blind Linear Transformation(ms)			Total(ms)		
		Min	Max	Std	Min	Max	Std	Min	Max	Std	Min	Max	Std
20	20	753	784	12.3333	240	257	41.4620	44	47	0.7263	1038	1088	16.9773
20	40	1129	1177	17.3433	361	382	5.9521	44	46	0.7263	1536	1603	23.1471
20	60	1505	1567	24.9469	480	514	10.0668	44	48	1.1790	2034	2127	34.7969
60	60	2257	2352	31.3503	717	767	11.6645	44	47	0.7681	3018	3162	42.5677
60	120	3389	3550	63.3628	1079	1159	23.8778	44	47	1.1169	4515	4756	86.4866
60	180	4516	4729	72.3795	1446	1527	24.3234	44	47	0.9000	6009	6294	94.7878
100	100	3765	3918	52.0802	1204	1273	16.0409	44	46	0.7348	5016	5234	65.2713
100	200	5647	5914	101.2305	1786	1930	40.1426	44	47	0.8646	7477	7882	139.7714
100	300	7526	7841	125.4932	2395	2588	53.3105	44	46	0.7399	9965	10464	175.8158

TABLE II
EXPERIMENT RESULTS OF THE PROPOSED SCHEME

As is shown from Fig. 1-(a)(b)(d), the growth of the execution time remains linear in all cases. This makes sense since the complexity of each encryption or decryption is only determined by the fixed key size. Thus the decryption or encryption time grows in proportion to the number of attributes. Fig. 1-(a) also indicates that most of the computation time is spent in Blind Transformation phase. This is true as the encryption is the most expensive part in our implementation. Pre-encrypting the attributes in the profile off-line could further improve the performance.

As Fig. 1-(c) shows, the Blind Linear Transformation phase in the protocol introduces low overhead, within $48ms$ in the transformation step. Thus, compared with the basic scheme, the advanced scheme is secure while incurs little overhead.

Using different security parameter l will give different performance since l increases the total vector size and the number of encryption/decryption operations. We present these 3 parameter settings to demonstrate a trade-off between security and efficiency. However, given the fact that even if the adversary has guessed one attribute correctly, he has no way to verify it and thus, setting $l = n$ is enough in most cases since in comparison with the number of attributes(normally 20 to 30 as in [5][4]), the $\frac{3}{ln} = \frac{3}{n^2}(l = n)$ is far less than a random guess with probability $\frac{1}{n}$. Thus it's secure enough in most cases. The efficiency of our scheme is better than some existing protocols such as the secure friend discovery in [4]. With $l = n$, our implementation out-performs [4] with 40% less running time (1.556s compared to 2.6s in [4]), despite the fact that [4] runs on Intel Core Duo P8600(2.4GHz), whose clock speed is faster than our i3-330m, and the simulation is a single thread task with no speed up provided by the Hyper-Threading technology in Core i3-330m.

Note that the computation overhead on verifiers is not measured in the simulation. But it's clear that the only task for the verifiers is to test whether two integers(no larger than 256 bit when using SHA-256) are equal. The transportation overhead and power consumption for them is negligible.

VI. CONCLUSION

In this work, we have developed a novel protocol that ensures the fairness and the privacy of privacy-preserving interest/profile matching process in mobile social networks. The experiment result substantiates the efficiency of our scheme, thus coped with the security analysis, we can conclude our work as secure and efficient. Our future work includes providing fine-grained interest/profile matching and investigating more issus on security and privacy in mobile social networks.

ACKNOWLEDGEMENT

This research is supported by National Natural Science Foundation of China (Grant No.61003218, 70971086, 61161140320, 61033014), Doctoral Fund of Ministry of Education of China (Grant No.20100073120065).

REFERENCES

[1] "foursquare," 2012. [Online]. Available: https://foursquare.com/
[2] "gowalla," 2012. [Online]. Available: http://gowalla.com/
[3] 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 *2010 International Conference on Distributed Computing Systems*, Jun. 2010, pp. 468–477.
[4] W. Dong, V. Dave, L. Qiu, and Y. Zhang, "Secure friend discovery in mobile social networks," in *Proceedings of IEEE INFOCOM 2011*. IEEE, Apr. 2011, pp. 1647–1655.
[5] 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*. IEEE, Apr. 2011, pp. 2435–2443.
[6] R. Lu, X. Lin, X. Liang, and X. Shen, "A secure handshake scheme with symptoms-matching for mhealthcare social network," *Mobile Networks and Applications*, pp. 1–12, 2010.
[7] M. Von Arb, M. Bader, M. Kuhn, and R. 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,*. Ieee, 2008, pp. 184–189.
[8] L. Kissner and D. Song, "Privacy-preserving set operations," in *Advances in Cryptology–CRYPTO 2005*. Springer, 2005, pp. 241–257.
[9] Q. Ye, H. Wang, and J. Pieprzyk, "Distributed private matching and set operations," *Information Security Practice and Experience*, pp. 347–360, 2008.
[10] M. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology-EUROCRYPT 2004*. Springer, 2004, pp. 1–19.
[11] E. D. Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear computational and bandwidth complexity," 2010.
[12] "perfectmatch," 2012. [Online]. Available: http://www.perfectmatch.com
[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology–EUROCRYPT'99*. Springer, 1999, pp. 223–238.
[14] R. Rivest, "Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer," *Unpublished manuscript*, 1999.
[15] "Paillier homomorphic cryptosystem (java implementation)," 2012. [Online]. Available: http://www.csee.umbc.edu/~kunliu1/research/Paillier.html
[16] D. E. Knuth, *The Art of Computer Programming volume 2:Seminumerical algorithms*. Addison Wesley, 1969.