2011 Ninth Annual International Conference on Privacy, Security and Trust

Privacy-Preserving Matchmaking For Mobile Social Networking Secure Against Malicious Users

Qi Xie and Urs Hengartner
Cheriton School of Computer Science, University of Waterloo
{q7xie,uhengart}@cs.uwaterloo.ca

***Abstract*—The success of online social networking and of mobile phone services has resulted in increased attention to mobile social networking. Matchmaking is a key component of mobile social networking. It notifies users of nearby people who fulfil some criteria, such as having shared interests, and who are therefore good candidates for being added to a user's social network. Unfortunately, the existing matchmaking approaches are troublesome from a privacy point of view. One approach has users' smartphones broadcast their owners' personal information to nearby devices. This approach reveals more personal information than necessary. The other approach requires a trusted server that participates in each matchmaking operation. Namely, the server knows the interests and current location of each user and performs matchmaking based on this information. This approach allows the server to track users.**

This paper proposes a privacy-preserving matchmaking protocol for mobile social networking that lets a potentially malicious user learn only the interests (or some other traits) that he has in common with a nearby user, but no other interests. In addition, the protocol is distributed and does not require a trusted server that can track users or that needs to be involved in each matchmaking operation. We present an implementation and evaluation of our protocol on Nexus One smartphones and demonstrate that the protocol is practical.

I. Introduction

In the last decade, the number of users of online social networking sites and of mobile phone services has skyrocketed. For example, the most popular online social networking site, Facebook, has more than 500 million active users, and more than 50% of its active users log on to Facebook at least once per day [12]. In terms of mobile phone services, there were 4.1 billion mobile cellular subscribers in total in March 2009 [22].

Mobile social networking brings these two fast-growing services together. On social networking sites, other than communicating with existing friends, people can find and make friends with other people who have similar interests, are from the same school or company, etc. In mobile social networking, users can find new friends as they do on traditional social networking sites, but there is an extra matching criterion: geographical distance between two users. More specifically, people walk around with their mobile phones and meet different people, known and unknown ones, every day. When two mobile phones are geographically nearby, a matchmaking operation takes place and detects common interests (or some other joint attribute) of the devices' owners. If a match is found, the devices notify their owners, who can immediately meet each other in person. For example, a mobile social networking user may find out that the person sitting next to her on the bus was her schoolmate or that her uptight boss likes the same comedy as she does. Mobile social networking is currently a hot topic and numerous matchmaking applications have been developed by different companies or research groups, such as MobiClique [23], looptmix [19], or Gatsby [14].

In general, there are two straightforward ways to implement matchmaking in mobile social networking. One way is for a device to broadcast its owner's profile information to the public, for example using Bluetooth. MobiClique [23] is an example of the applications that take this approach. MobiClique users download their profile information from Facebook to their device and send this information to any Bluetooth device nearby. After receiving a piece of profile information, a device performs a matching between the received profile information and its owner's profile information and decides whether the other party is of the owner's interest according to the matching result. This approach is risky because it leaks users' private information to anyone in the users' proximity. The other way to implement matchmaking in mobile social networking is to introduce a trusted server for the matchmaking operation. looptmix [19] and Gatsby [14] are examples that use this approach. Here, the server stores all users' personal information and tracks their location. The server informs two users if they are nearby and could become friends based on their requirements. This approach has the limitation that it requires the server to be always available in order to find a friend. From a privacy point of view, the server learns all users' personal information, all pairs of users who meet each other, and the locations of all users.

Our matchmaking protocol for mobile social networking is based on the following two principles:

1) *Ensure privacy by making users reveal no unnecessary private information to other users in order to find new nearby friends.* Two nearby users learn only personal information about the other person that they have in common and no other information.
2) *Avoid a trusted server that participates in each matchmaking operation.* Users do not need Internet access to

find friends, and users do not need to worry about being tracked by a server.

The contributions of our paper are as follows:

1) We propose a matchmaking protocol for mobile social networking that is secure against malicious users. In this paper, we focus on interest matching as an example, but the protocol can be easily adapted to find friends who graduated from the same school, worked for the same company, etc.
2) We implement the protocol and evaluate its performance. Our performance results demonstrate that the protocol is practical.

The remainder of this paper is organised as follows: In Section II, we discuss related work in more detail. We then present our system and threat model in Section III. In Section IV, we present our matchmaking protocol. Our description of the implementation and evaluation of the protocol is in Section V.

II. Related Work

We first discuss previous work on matchmaking for mobile social networking, then we review cryptographic protocols for matchmaking.

A. Mobile Social Networking Applications

Previous research on mobile social networking has suggested matchmaking applications, but has ignored privacy concerns. Social serendipity [11] deploys a trusted server that contains users' profiles and user-defined matchmaking preferences. Based on this information, the server computes the similarity of nearby users' profiles. Users will be notified when they are nearby and the similarity of their profiles exceeds a threshold. Obviously, the server has to be involved in every matchmaking operation, which is a limitation of this system. In addition, the server is able to learn which two users are nearby and have matching interests. Furthermore, the server can track a user's location. MobiClique [23] improves social serendipity by taking the server away from the matchmaking operation. This system has a server, Facebook, assign identifiers to the users. It then allows users to store their profile information on their mobile devices and exchange their profiles with their neighbours using Bluetooth. New friendships are established based on the received user profiles. This system does not take malicious users into account. For example, users' profile information is exchanged in plaintext, which could be intercepted by anybody within range, and this information could be used for malicious purposes, such as context-aware attacks. Also, users could be impersonated, because the system does not have a method to validate identifiers. In our protocol, users whose interests do not match do not learn any information other than that their interests do not match. In addition, we use signed identifiers to identify users, which avoids impersonation attacks.

SmokeScreen [6] addresses "missed connections", the problem that users who met before want to talk to each other again at a later time and need to convince each other that they were at the same location at the same time. SmokeScreen prevents users from revealing their real identifiers to undesired people. When users meet, they exchange opaque identifiers rather than their real identifiers. They record the time and position when they receive the opaque identifiers and then contact each other at a later time through a trusted third party, a broker, which is able to read the opaque identifiers. Again, this approach requests a trusted server for every matchmaking operation. Also, because no information is shared by the participants before their identifiers are exchanged, there is no incentive for the participants to exchange their real identifiers (they know nothing else about the other party). In SmokeScreen, the broker knows who is interested in solving whose opaque identifier. SMILE [20] is an advanced version of SmokeScreen that further guarantees that the server cannot learn which pairs of users encounter each other. Again, the server has to participate in each matchmaking operation. The system still does not take personal information, such as interests into account, but only location and time. Our protocol allows users to establish friendship as soon as they meet each other without requiring a trusted server that is involved in each matchmaking operation, and we provide more matching criteria.

B. Matchmaking Protocols

In this section, we discuss several cryptographic protocols that seem suitable to solve the matchmaking problem.

The matchmaking problem is an instance of the private set intersection problem, where two parties, each having a set of elements, want to compute the intersection of their sets such that no party can learn information other than the intersecting elements. Most solutions for the private set intersection problem (e.g., [3], [13], [15], [16], [25]) let each party freely choose the elements in her set. These solutions are not appropriate for our matchmaking problem because they allow a malicious user to guess a set of plausible interests that the other user might have. In the extreme case, the user chooses all possible interests as her input set.

To address this problem, Camenisch and Zaverucha [5] force users to use inputs that are certified by a trusted third party so that the user can no longer freely choose her inputs for each protocol run. The protocol is based on advanced cryptographic techniques, such as CL-signatures [4] used for signing set elements and zero-knowledge proofs for proving correct processing. This makes the computational overhead of the protocol high and not suited for smartphones, which are our target platform. Namely, the protocol requires $O(k^2)$ modular exponentiations (k is the size of an input set), whereas our protocol requires only $O(k)$ modular exponentiations. Camenisch and Zaverucha's protocol provides untraceability as an additional benefit, that is, a user appears as a different user for each run of the protocol. However, untraceability is not a desired feature in our application scenario, where users are physically close and can recognise each other anyway based on their physical features. Moreover, since untraceability makes it impossible to recognise users with whom a user has already run the matchmaking protocol, the users would unnecessarily re-run the protocol whenever they meet again, which wastes battery and computation resources.

De Cristofaro and Tsudik [9] and De Cristofaro et al. [8] present protocols for "Authorized Private Set Intersection", where a client and a server run a private set intersection protocol such that only the client learns the elements in the intersection and where the client's inputs to the protocol are certified by a trusted third party, but not the server's inputs. The protocols require only a linear number of modular exponentiations. At first sight, it looks tempting to apply these protocols to matchmaking in mobile social networking. However, there are two obstacles. First, the protocols are one-way only, whereas we want both parties to learn the elements in the intersection. Whereas it is possible to run a protocol twice, with reversed roles in the second run [9], the fact that only one of the two parties' inputs are certified lets a party use different input values in each run. For example, a malicious server that wants to learn a client's interests but is unwilling to reveal his own interests can lie about his interests in the first run since the server's inputs are not certified. In the second run, where the roles are reversed, the (former) server will now have to use his certified inputs, but the (former) client will not learn anything but that the server used some certified inputs, so the (former) client will not get suspicious. Our protocol is two-way, and both parties' inputs need to be certified. Second, the existing protocols do not bind the certified inputs to an identity. Therefore, malicious users could collude with each other and exchange each other's certified inputs. This way a malicious user can use the certified inputs that look most plausible given a victim in a particular run of the protocol. In our protocol, certified inputs are bound to an identity, which defends against the exchange of these inputs among users.

Agrawal et al. [1] propose a protocol using a commutative encryption function for private intersection problems. A commutative encryption function has the property: $E_{k1}(E_{k2}(P))$ = $E_{k2}(E_{k1}(P))$. Either user learns that $P = P'$ only when $E_{k1}(E_{k2}(P)) = E_{k2}(E_{k1}(P'))$, but neither of them could learn the other party's information outside of the intersection because of lacking necessary key information. Agrawal et al. suggest the power function, $f_e(x) = x^e \bmod p$, as an example of a commutative function. The security of their protocol is based on the Decisional Diffie-Hellman hypothesis (DDH). If we assume that the size of each user's set is k, this protocol requires k modular exponentiations for each user. We improve and implement this approach in our matchmaking protocol. More details about this protocol and our extensions are provided in Section IV.

Agrawal et al. use a commutative encryption function that is based on public-key cryptography, which is more expensive than symmetric-key cryptography. Therefore it is tempting to solve the private set intersection problem by using simple symmetric-key approaches. However, this work is often flawed. For example, Shundong et al. [26] propose to use the XOR operation as the symmetric commutative function, which sharply reduces the computational overhead. However, this solution is broken [27].

We have seen that existing matchmaking applications either require a trusted third party in each matchmaking operation or do not consider protecting users' personal information. Our protocol needs a trusted third party only in the setup phase for certifying a user's interests, but does not need it in each matchmaking operation. The protocol is based on asymmetric key-based cryptography.

Concurrent with this work, several other privacy-preserving matchmaking protocols have been developed [7], [10], [18]. However, none of them defends against malicious users.

III. System and Threat Model

Our system model consists of users and their mobile devices. Each device knows its owners' interests (see below for details) and the goal is to find nearby devices whose owners have at least one matching interest. Users might be malicious and try to learn other users' interests, without having these interests themselves.

According to our design principles in Section I, there should be no trusted server that is involved in each matchmaking operation. We have two nearby devices communicate directly with each other over Bluetooth, which is a communication channel for devices in proximity. Bluetooth is attractive since compared to WiFi, it uses less power.

Our protocol requires two third parties for setup purposes: an *identity signer* and a *personal interest signer (PIS)*. Note that these parties are not involved in the actual matchmaking and therefore cannot track users and do no need to be available continuously. The identity signer assigns an identity certificate to each user and guarantees that one user is assigned to only one identifier. For example, the identity signer could be a Facebook application that issues an identity certificate to a Facebook user. Facebook's Terms of Service prohibit users from creating more than one profile, and Facebook is known to disable fake accounts [24]. The PIS signs a user's interests. To use our matchmaking protocol, a user must obtain signed certificates for her interests from the PIS. Similar to the identity signer, the PIS could be a Facebook application because many people have already submitted their interests to Facebook. We choose Facebook in our implementation because it is used by many people, but it is possible to build additional implementations for other platforms.

Our matchmaking protocol prevents attackers from executing the following attacks:

- Learning a user's interests without getting caught for cheating unless an attacker actually has the same interest(s), as certified by the PIS.
- Exploring a user's interests by including all possible or a large number of popular elements in the attacker's interest set. The PIS puts a limit on the number of interests that it certifies for each user to avoid brute-force attacks. This can be a limitation for users that have more interests than this threshold, but we think that it is a reasonable trade-off when it comes to defending against brute-force attacks. Ideally, we could drop the threshold and ensure instead that a user actually has the certified interests but this

is difficult to do in an automated way in practice. The certificates have a limited lifetime, which allows users to update their interests over time.

- Exploring a user's interests by exchanging (or maybe stealing) signed certificates obtained from the PIS with (or from) other users and then choosing the certificates that are input to a particular protocol run by guessing a set of interests that the victim with whom the protocol is run is most likely to have. The certificates generated by the PIS are bound to a particular user, which defends against certificate exchanging (or stealing) attacks.
- Impersonating other users. Each user creates an asymmetric key pair and uses the hash value of the public key as her user id. The identity signer will include a user's user id in her identity certificate. When two users run a matchmaking protocol, they authenticate each other using their public keys and may negotiate a session key.
- Actively or passively eavesdropping the communication between any two users. Sensitive information is encrypted by the session key that the two users established. Exchanged information is also authenticated with digital signatures for non-repudiation purposes.

We do not consider the following threats and make the following assumptions:

- Users keep their private keys safe, so that malicious users cannot steal their private keys to impersonate them.
- Neither the identity signer nor the PIS is compromised by attackers.
- Users trust the person with whom our protocol finds the user to have a common interest not to disclose this interest. Any matchmaking protocol needs this assumption. If a user does not feel comfortable with it, she should not run the protocol.
- Users are going to finish running a matchmaking protocol once they start it. Otherwise, there might be information asymmetry, where one user learns more information than the other user does. This is an instance of the fair-exchange problem, which is hard to solve without a third party [2]. In our protocol, users first authenticate each other. This way, if a user prematurely ends the protocol, the other user can report this misbehaviour to the PIS, which will ultimately refuse to renew the user's certificates.
- We do not consider tracking attacks. Our protocol requires users to use consistent user ids. This allows people to recognise each other, which makes tracking attacks possible. However, our protocol is based on Bluetooth, which is a short-range communication technique. Therefore, an attacker can track a user only if the attacker is nearby the user. In other words, a tracking attack requires the attacker to physically follow the user. A user is vulnerable to this kind of attack, no matter whether she is using our matchmaking protocol or not. Even if we allowed users to use different user ids, they would still be physically trackable. In addition, as mentioned in Section II, defending against tracking attacks could lead to a user repeatedly running the matchmaking protocol with the same user again and again, which wastes battery and computation resources.
- We do not consider the number of interests that a user gives as input to the matchmaking protocol to be sensitive. Whereas there are protocols that allow this number to be hidden [3], to the best of our knowledge, there are no protocols that both have this feature and support certified inputs.

IV. Matchmaking Protocol

A. Agrawal et al.'s Protocol

Our matchmaking protocol is based on the protocol by Agrawal et al. [1] discussed in Section II-B, but we extend the protocol to defend against attacks. Let us review Agrawal et al.'s protocol first, which is shown in Figure 1. We refer to Agrawal et al.'s paper for a formal proof of security.

Assume the users are Alice and Bob. Alice has a set S_A, and Bob has a set S_B. Alice and Bob randomly choose a secret key, k_A and k_B, respectively. In step 1, both Alice and Bob exponentiate each element of their set with their secret key. In the remainder of this paper, we have a^e represent $a^e \mod p$, where p is a safe prime. p is a safe prime if and only if p is a prime number and $(p-1)/2$ is also a prime number. $h_p()$ represents a cryptographic hash function that has the quadratic residues modulo p as its range. In step 2, Alice sorts the exponentiated elements lexicographically and then sends them to Bob. In step 3, Bob performs the corresponding operation. He also exponentiates the values received from Alice and returns them. In step 4, Alice creates $\langle s_i, (h_p(s_i)^{k_A})^{k_B}\rangle \ \forall s_i \in S_A$ by replacing $h_p(s_i)^{k_A}$ in the pairs that she received from Bob in step 3 with corresponding s_i, $\forall s_i \in S_A$. In step 5, Alice computes the intersection between the set consisting of the second element of the pairs $\langle s_i, (h_p(s_i)^{k_A})^{k_B}\rangle \ \forall s_i \in S_A)$ and set Z. The first element of pairs whose second element is in the intersection forms the set of elements that Alice and Bob have in common.

Applying this protocol to the matchmaking problem raises several challenges. First, the protocol does not use certified set elements, which allows an adversary to freely choose the inputs to the protocol. Second, only Alice learns the intersecting elements, and she could lie to Bob about the result. Third, there is no guarantee that Bob will not reorder the list formed by the second element of the pairs $\langle h_p(s_i)^{k_A},\ (h_p(s_i)^{k_A})^{k_B}\rangle \ \forall h_p(s_i)^{k_A}$ in step 3. This will result in Alice mis-identifying an intersecting element. If Alice reported the index of the intersecting pair to Bob, Bob could still correctly identify the intersecting element.

B. Our Matchmaking Protocol

We improve this protocol and make it fit our requirements. Our protocol has three phases: the initial phase, the interest signing phase, and the matchmaking phase.

Alice's inputs: set S_A, secret key k_A
Bob's inputs: set S_B, secret key k_B

1) Alice computes $h_p(s_i)^{k_A}$ $\forall s_i \in S_A$.
   Bob computes $h_p(s_i)^{k_B}$ $\forall s_i \in S_B$.
2) Alice $\xrightarrow{\forall s_i \in S_A\colon\, h_p(s_i)^{k_A}}$ Bob
3) Alice $\xleftarrow{\forall s_i \in S_B\colon\, h_p(s_i)^{k_B}}$ Bob
   Alice $\xleftarrow{\forall h_p(s_i)^{k_A} \text{ received from Alice in step 2: } \langle h_p(s_i)^{k_A},\, (h_p(s_i)^{k_A})^{k_B} \rangle}$ Bob
4) Alice computes set $Z = (h_p(s_i)^{k_B})^{k_A}$ $\forall h_p(s_i)^{k_B}$ received from Bob in step 3.
   Alice creates list $Y = \langle s_i, (h_p(s_i)^{k_A})^{k_B} \rangle$ $\forall s_i \in S_A$.
5) Alice intersects set Z and the set formed by the second element of the pairs in list Y. The set that consists of the first element of the pairs in Y whose second element is in the intersection is identical to $S_A \bigcap S_B$.

Fig. 1. Agrawal et al.'s Private Set Intersection Protocol [1].

1) Initial Phase: This phase takes place between the identity signer and a user (e.g., Alice). As mentioned before, the identity signer guarantees that one user is assigned to only one identifier. Alice sends personal information, such as her name or her Facebook profile identifier, to the identity signer. Alice also generates an RSA key pair and sends the public key to the identity signer. The identity signer identifies and authenticates Alice and issues an identity certificate to her. The certificate includes Alice's RSA public key. Alice will later use the hash value of her public key as her user id. Strictly speaking the initial phase needs to take place only once for each user, though it might have to be repeated when a user's identity certificate expires.

2) Interest Signing Phase: This phase takes place between the personal interest signer (PIS) and a user (e.g., Alice). The PIS generates a safe prime, p, the first time when it starts. When a user creates a name for a new interest (no other user has submitted the name yet), the PIS chooses a quadratic residue modulo p as the id of this interest. In more detail, Alice submits the names of her interests, along with her identity certificate and her user id to the PIS. The PIS puts a limit on the number of interests that Alice can submit to avoid brute-force attacks. The PIS then makes sure that Alice did not misbehave before (based on other users' complaints). Here let us assume that Alice is an honest user, and no other users have complained about her. We denote the ids of Alice's interests by X_i $\forall i \in (0, m]$ (assume that Alice has m interests). To determine X_i for a particular interest, the PIS first uses SHA-1 to compute the hash value of the name of the interest and then computes the square modulo p of the hash value. The PIS randomly generates an exponent, a (a is in the range of $[1, q-1]$, where $q = (p-1)/2$) for Alice and computes X_i^a $\forall i \in (0, m]$ and $sign_{\text{PIS}}(A\_ID \| X_i^a)$ $\forall i \in (0, m]$. ($sign_{\text{PIS}}()$ denotes a signature created by the PIS. A_ID denotes Alice's user id.) The *PIS* sends X_i $\forall i \in (0, m]$, a, p, X_i^a $\forall i \in (0, m]$, $sign_{\text{PIS}}(A\_ID \| X_i)$ $\forall i \in (0, m]$, and $sign_{\text{PIS}}(A\_ID \| X_i^a)$ $\forall i \in (0, m]$ back to Alice.

The interest signing phase takes place whenever a user's interests change (or when a signature expires), though the PIS must limit the frequency at which users can perform such changes to defend against brute-force attacks.

3) Matchmaking Phase: Suppose that Alice has interests: $X_1, X_2, \ldots, X_m$ (m is the size of Alice's set), and Bob has interests: $Y_1, Y_2, \ldots, Y_n$ (n is the size of Bob's set). $h()$ denotes a cryptographic hash function such as SHA-1. Let us assume that in this protocol, the party who initiates the communication reports the result first. We also assume that Alice and Bob agree on a value g. Note that $user\_ID = h(public\_key)$. Figure 2 shows our matchmaking protocol. Before running this protocol, Alice and Bob should pair their Bluetooth devices and run a signature-based authentication protocol (e.g., [21, p. 404]) to identify and authenticate each other (not shown in the figure). Note that steps 9-14 are executed only if there is at least one match.

In steps 1-4, Alice and Bob exchange their exponentiated values, as received from the PIS, and the corresponding signatures to ensure authenticity of these values. Alice and Bob sign their messages to ensure non-repudiation in case misbehaviour is detected later. Step 5 is required for cheating detection purposes (see next section for details). Alice sends a commitment to her computed values to Bob, where R is a random number of fixed length that ensures that the commitment leaks no information. In step 6, Bob reports his computed results to Alice for matching. In step 7, Alice reports her computed results to Bob, so Bob can perform the matching. In addition, Alice needs to reveal the values that she committed to in step 5. Alice and Bob both compute the interests they have in common in step 8. If there are no matching interests, the protocol is terminated.

There is no guarantee that Alice will pair $\left(Y_i^b, \left(Y_i^b\right)^a\right)$ $\forall i \in (0, n]$ correctly in step 7. For example, Alice can pair Y_i^b with $\left(Y_j^b\right)^a$ for $i \neq j$. Later, we will show that Alice could take advantage from mispairing the data. Bob could also do the same trick to Alice. As a result, we require Alice and Bob to provide the signatures of any matched interests next. Namely, in steps 9-14, Alice and Bob exchange the signatures issued by the *PIS* of the interests that they have in common to prove that they really have the interests. However, they

1) Alice $\xrightarrow{(\forall i\in(0,m]:\ X_i^a \| sign_{\mathrm{PIS}}(A\_ID \| X_i^a)) \| sign_{\mathrm{Alice}}(\text{entire message})}$ Bob
2) Alice $\xleftarrow{(\forall i\in(0,n]:\ Y_i^b \| sign_{\mathrm{PIS}}(B\_ID \| Y_i^b)) \| sign_{\mathrm{Bob}}(\text{entire message})}$ Bob
3) Alice and Bob verify all signatures.
4) Alice computes $\left(Y_i^b\right)^a$ $\forall i \in (0,n]$ and Bob computes $\left(X_i^a\right)^b$ $\forall i \in (0,m]$.
5) Let $resp_A = (Y_1^b)^a \| \ldots \| (Y_n^b)^a \| R$
   Alice $\xrightarrow{h(resp_A) \| sign_{\mathrm{Alice}}(B\_ID \| h(resp_A))}$ Bob
6) Let $resp_B = (X_1^a)^b \| \ldots \| (X_m^a)^b$
   Alice $\xleftarrow{(\forall i\in(0,m]:\ (X_i^a,(X_i^a)^b)) \| sign_{\mathrm{Bob}}(A\_ID \| resp_B \| (\forall i\in(0,m]:\ X_i^a))}$ Bob
7) Alice $\xrightarrow{(\forall\ i\in(0,n]:\ ((Y_i^b,(Y_i^b)^a)) \| R \| sign_{\mathrm{Alice}}(B\_ID \| resp_A \| (\forall i\in(0,n]:\ Y_i^b))}$ Bob
8) Bob gets Y_i $\forall Y_i \in \left(Y_i^b\right)^a \bigcap \left(X_i^a\right)^b$. If the intersection is empty, Bob terminates the protocol. Alice gets X_i $\forall X_i \in \left(Y_i^b\right)^a \bigcap \left(X_i^a\right)^b$. If the intersection is empty, Alice terminates the protocol.
9) Alice $\xrightarrow{g^a}$ Bob
10) Alice $\xleftarrow{g^b \| sign_{\mathrm{Bob}}(g^a \| g^b \| A\_ID)}$ Bob
11) Alice $\xrightarrow{sign_{\mathrm{Alice}}(g^b \| g^a \| B\_ID)}$ Bob
12) Alice computes $k = (g^b)^a$ and Bob computes $k = (g^a)^b$.
13) Alice $\xrightarrow{\forall i\in(0,m']:\ E_k(X_i \| sign_{\mathrm{PIS}}(A\_ID \| X_i))}$ Bob (assuming there are m' common interests)
14) Alice $\xleftarrow{\forall i\in(0,m']:\ E_k(Y_i \| sign_{\mathrm{PIS}}(B\_ID \| Y_i))}$ Bob

Fig. 2. Our Matchmaking Protocol.

should not send this information in plaintext, because the information could be eavesdropped. Therefore, in steps 9-12, Alice and Bob negotiate a secret using authenticated Diffie-Hellman [17] to encrypt the sensitive messages. We could have combined steps 9-12 with the signature-based authentication protocol executed before the matchmaking protocol. However, computing modular exponentiations, as required by the Diffie-Hellman protocol, is relatively expensive, so we make sure that we execute this protocol only if two users actually have at least one common interest. After step 14, both Alice and Bob verify the signatures and record the messages if the other party fails to provide correct interest ids and signatures. In this case, the misbehaving party should be reported to the *PIS*, together with the recorded information. Even in case the signatures can be verified and the interest ids are correct, the recipient should still record the other party's messages at a given probability and report them at a later time to the *PIS* to check if the other party cheated or not. We explain why this is necessary in the next section.

C. Security Analysis

1) Passive Attacks: We first look at passive attacks, where an attacker tries to infer sensitive information from observed protocol messages. For simplicity, we focus on an attack on messages sent by Alice, but our analysis can be easily applied to Bob's messages. Agrawal et al. prove that given the Decisional Diffie-Hellman hypothesis (DDH), $\langle X_i, f_e(X_i), Y_j, f_e(Y_j)\rangle$ for fixed values of i and j, with $f_e(x) = x^e$, is indistinguishable from $\langle X_i, f_e(X_i), Y_j, Z\rangle$, when e is not given. Further, they conclude that for polynomial t and k,

$$\begin{pmatrix} X_1 & \ldots & X_t & X_{t+1} & \ldots & X_k \\ f_e(X_1) & \ldots & f_e(X_t) & f_e(X_{t+1}) & \ldots & f_e(X_k) \end{pmatrix}$$

is indistinguishable from

$$\begin{pmatrix} X_1 & \ldots & X_t & X_{t+1} & \ldots & X_k \\ f_e(X_1) & \ldots & f_e(X_t) & Z_{t+1} & \ldots & Z_k \end{pmatrix}.$$

These properties give us the following results:

1) Bob cannot map X_i^a back to X_i if he does not know the value of a, which is known only to Alice. Proof: if Bob could map X_i^a back to X_i, Bob could compute $f_a^{-1}(Z)$, and by checking whether $f_a^{-1}(Z) = Y_j$ or not, Bob can distinguish $\langle X_i, f_a(X_i), Y_j, f_a(Y_j)\rangle$ and $\langle X_i, f_a(X_i), Y_j, Z\rangle$. This makes sure that before Alice sends Bob the computed result in step 7, Bob is not able to learn any of Alice's interests. Due to the same reason, an eavesdropper observing X_i^a cannot learn X_i.
2) Given the values of $X_1, f_a(X_1), \ldots, X_t, f_a(X_t), X_{t+1}, f_a(X_{t+1}), \ldots, X_k, f_a(X_k)$, Bob cannot compute the value of a. Proof: assume that Bob could compute the value of a, so he is able to compute $f_a(X_k)$, and by checking if $f_a(X_k) = Z_k$, Bob could distinguish the two matrices given above. This guarantees that Bob cannot obtain Alice's exponent, a. As a result, even if Bob knows the id V of an interest that he does not have, he cannot compute V^a. Therefore, he cannot detect if Alice has this interest by checking if V^a equals X_i^a $\forall i \in (0,m]$. The same applies to an eavesdropper observing the values.

In steps 13 and 14, Alice and Bob exchange their signed interest certificates for all their matching interests. These certificates are sent across an encrypted channel, so a passive eavesdropper cannot learn Alice's or Bob's interests.

2) Active Attacks involving Non-Common Interests: We look at active attacks next. Note that Alice and Bob authenticate each other and that protocol messages are signed, which allows the detection of impersonation or modification attacks by active eavesdroppers. Therefore, we focus on active attacks by Alice or Bob. We first study scenarios where Alice or Bob misbehave to learn interests other than the ones that the two of them have in common. As it turns out, if one of them misbehaves in such a way, our protocol allows the other party to detect this cheating.

By sending out $\{Y_1^b, \ldots, Y_n^b\}$, and getting back $\{X_1^a, \ldots, X_m^a\}$ and $\{(Y_1^b, (Y_1^b)^a), \ldots, (Y_i^b, (Y_i^b)^a)\}$, Bob can learn only whether or not Alice has a number of elements in set $\{Y_1, \ldots, Y_n\}$. According to the above two properties, if and only if $(Y_i^b)^a = (X_i^a)^b$, Bob can learn which X_i from Alice matches one of his Y_i by performing step 8. Otherwise, it is impossible for him to get the value of X_i. Note that Bob cannot be certain which interests Alice really shares with him until they exchange the signatures signed by the *PIS* including the interest ids. Namely, Alice could pair Y_j^b with $\left(Y_i^b\right)^a$ for $j \neq i$ in step 7, and Bob would think Alice has interest Y_j instead of Y_i if $\left(Y_i^b\right)^a$ is in the intersection. Bob would send Y_j to Alice. As a result, Alice is able to gain extra information without being detected. Exchanging the signatures of the signed interests will detect this attack. As a result, it is necessary for both users to exchange the signatures of the ids of the common interests. If a user cannot provide a valid signature signed by the *PIS* for all her common interests, the other user should send the protocol transcript to the *PIS*. Because Alice receives messages from Bob that are symmetric to the messages that she sends to Bob, she is also unable to learn extra interests.

Alice and Bob cannot use interests not signed by the *PIS* for them. For instance, Alice has to provide $sign_{\mathrm{PIS}}(A\_ID \| X_i^a)\ \forall i \in (0, m]$ for all elements in her set to prove that she really is assigned this interest before the matching. Unlike the symmetric key-based based commutative functions, it is not possible to find $X'^{a'} = X^a$ because of the way we create the X and X' values and because of the discrete logarithm problem. This guarantees that Alice can use $sign_{\mathrm{PIS}}(A\_ID \| X^a)$ only to prove her ownership of interest X but for nothing else.

Alice and Bob cannot get useful information by replaying other users' responses. They have to run a signature-based authentication protocol before executing the matchmaking protocol. This prevents Alice and Bob from using signatures created by or assigned to other users.

3) Active Attacks involving Common Interests: Let us look next at active attacks where Alice tries to prevent Bob from learning an interest that the two of them have in common (or vice versa). Our protocol provides methods to detect this cheating.

In the first attack, Alice wants to find new friends, but if there are multiple common interests, she wants to reveal only one interest because this is still enough to find a new friend. If we removed step 5 from our protocol, Alice could easily achieve this goal. Assume Alice and Bob have two interests in common. After Alice receives $(X_i^a)^b\ \forall i \in (0, m]$ from Bob in step 6, she knows that they have two common interests. Alice can lie and return the correct value of $(Y_i^b)^a$ for only one of the two matched interests. In this case, Alice finds a new friend and learns more information than Bob does. However, this attack is not possible in our protocol due to the commitment in step 5. Alice has to execute this step honestly since she does not know which interests they have in common at this step. As a result, she has to report $(Y_i^b)^a$ and R correctly to Bob in step 7, since it is hard to find $h(x') = h(x)$ for $x' \neq x$. Otherwise, Bob will detect her malicious behaviour immediately. Since Bob detects the common interests after Alice does, we do not have to introduce a commitment step for Bob.

In the second attack, Alice only wants to learn Bob's interests, but does not want to find a new friend. Therefore, in step 5, she sends Bob a commitment to some random values, instead of a commitment to $(Y_i^b)^a\ \forall i \in (0, n]$. Bob will learn of this misbehaviour immediately if Alice refuses to run step 7 and will report the misbehaviour to the PIS. If enough users report Alice's malicious behaviour, the PIS is not going to renew Alice's signatures. If Alice does run step 7, but returns the random values to Bob, instead of $(Y_i^b)^a\ \forall i \in (0, n]$, Bob is unable to detect this cheating since he cannot map $(Y_i^b)^a$ back to Y_i^a. However, Alice takes the risk that Bob records the protocol messages and reports them to the *PIS* later. (As we mentioned before, both users should record the protocol messages at a given probability, so that at a later time they can submit them to the *PIS* to check if the other party cheated or not). In this case, Bob has the signed messages from Alice, so the *PIS* is certain that Alice cheated and stops renewing the signatures of her interests. In the same way, Bob can detect if Alice misbehaves in step 6. Note that even if this attack succeeded and did not get reported to the PIS, the misbehaving party could learn only interests that the two parties have in common and not just any interest of the other party. An alternative to recording the protocol messages probabilistically and reporting them to the PIS is using zero-knowledge proofs that prove that a $(Y_i^b)^a$ was computed on the Y_i^b revealed in step 2 and that the value of a used in this computation was identical to the value of a in X_i^a revealed in step 1 (without revealing a). However, these zero-knowledge proofs are expensive and would increase the computation (and communication) overhead, which is why we choose a probabilistic detection method instead.

V. Implementation and Evaluation

We implemented our matchmaking protocol on two Google Nexus One smartphones, which have a 1 GHz processor and run Android 2.2. The two devices act as the mobile devices owned by two users. We used the Android SDK to implement

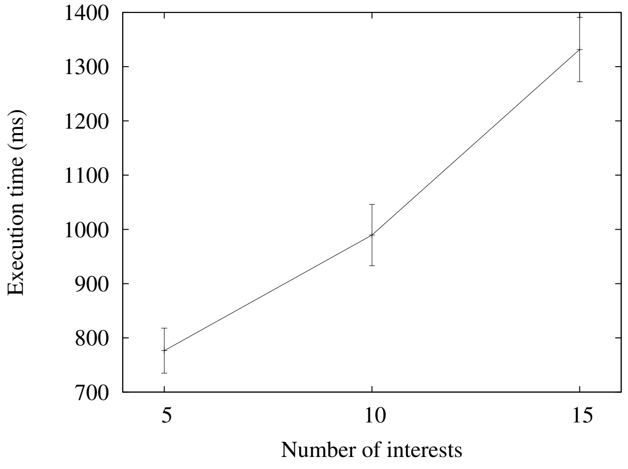

Fig. 3. Execution Time of the Matchmaking Protocol.

our matchmaking protocol. Our implementation of the *PIS* includes an application running on an application server, a web interface, and a backend database. The application issues signatures for users' interests and is implemented in Java. The web interface is implemented as a Facebook application using PHP and HTML. Users request and obtain their signatures through this Facebook application. The backend database stores interest information. We did not implement the id signer, and we assume that the users already hold valid identity certificates. We choose a prime p of length 1024 bits and use RSA with a modulus length of 1024 bits for the signatures.

The execution time of the protocol is shown in Figure 3. The x-axis shows the number of interests that each user has. There is one common interest for the five and ten interest case and six common interests for the 15 interest case. (We find that the number of common interests brings little difference to the performance results.) Every data point is the average of ten experiments. We also show the standard deviation. The time to perform a modular exponentiation is between 20 and 30 ms. For each interest, a device needs to perform exactly one modular exponentiation. The average execution time ranges from 776 ms (5 interests) to 1,332 ms (15 interests).

VI. Conclusions and Future Work

We presented a matchmaking protocol that preserves users' interest information from unnecessary leaks in mobile social networking applications. Our implementation and evaluation of the protocol show that the protocol is practical on current smartphones.

In terms of future work, we would like to investigate how real people use our matchmaking protocol to learn more about the usefulness of mobile social networking applications in general.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. This work is supported by a Google Research Award and by the Natural Sciences and Engineering Research Council of Canada.

References

[1] R. Agrawal, A. Evfimievski, and R. Srikant. Information Sharing Across Private Databases. In *Proc. of SIGMOD 2003*, pages 86–97.
[2] N. Asokan, V. Shoup, and M. Waidner. Optimistic Fair Exchange of Digitial Signatures. In *Proc. of EUROCRYPT'98*, pages 591–606.
[3] G. Ateniese, E. De Cristofaro, and G. Tsudik. (If) Size Matters: Size-Hiding Private Set Intersection. In *Proc. of PKC'11*, pages 156–173.
[4] J. Camenisch and A. Lysyanskaya. A Signature Scheme with Efficient Protocols. In *Proc. of Security in Communication Networks 2002*, pages 268–289.
[5] J. Camenisch and G. M. Zaverucha. Private Intersection of Certified Sets. In *Proc. of Financial Cryptography 2009*, pages 108–127.
[6] L. P. Cox, A. Dalton, and V. Marupadi. SmokeScreen: Flexible Privacy Controls for Presence-Sharing. In *Proc. of MobiSys '07*, pages 233–245.
[7] E. De Cristofaro, A. Durussel, and I. Aad. Reclaiming Privacy for Smartphone Applications. In *Proc. of PerCom 2011*, pages 84–92.
[8] E. De Cristofaro, J. Kim, and G. Tsudik. Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model. In *Proc. of Asiacrypt'10*, pages 213–231.
[9] E. De Cristofaro and G. Tsudik. Practical Private Set Intersection Protocols with Linear Computational and Bandwidth Complexity. In *Proc. of Financial Cryptography 2010*, pages 143–159.
[10] W. Dong, V. Dave, L. Qiu, and Y. Zhang. Secure Friend Discovery in Mobile Social Networks. In *Proc. of Infocom 2011*.
[11] N. Eagle and A. Pentland. Social Serendipity: Mobilizing Social Software. *IEEE Pervasive Computing*, 4(2):28–34, 2005.
[12] Facebook. Statistics. http://www.facebook.com/press/info.php?statistics. Accessed March 2011.
[13] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient Private Matching and Set Intersection. In *Proc. of EUROCRYPT 2004*, pages 1–19.
[14] Gatsby. http://meetgatsby.com/. Accessed March 2011.
[15] S. Jarecki and X. Liu. Fast Secure Computation of Set Intersection. In *Proc. of SCN 2010*, pages 418–435.
[16] L. Kissner and D. Song. Privacy-Preserving Set Operations. In *Proc. of CRYPTO 2005*, pages 241–257.
[17] H. Krawczyk. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols. In *Proc. of CRYPTO 2003*, pages 400–425.
[18] M. Li, N. Cao, S. Yu, and W. Lou. FindU: Privacy-Preserving Personal Profile Matching in Mobile Social Networks. In *Proc. of Infocom 2011*.
[19] looptmix. http://www.looptmix.com/. Accessed March 2011.
[20] J. Manweiler, R. Scudellari, and L. P. Cox. SMILE: Encounter-Based Trust for Mobile Social Services. In *Proc. of CCS 2009*, pages 246–255.
[21] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.
[22] MOCOM2020. 4.1 Billion Mobile Phone Subscribers Worldwide. http://www.mocom2020.com/2009/03/41-billion-mobile-phone-subscribers-worldwide/. Accessed March 2011.
[23] A-K Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot. Mobiclique: Middleware for Mobile Social Networking. In *Proc. of WOSN'09*.
[24] ReadWriteWeb. Facebook Confirms Bug Disabled User Accounts, Fix In Progress. http://www.readwriteweb.com/archives/female_facebook_users_take_to_twitter_complain_of.php. Accessed March 2011.
[25] Y. Sang and H. Shen. Efficient and Secure Protocols for Privacy-Preserving Set Operations. *ACM Transactions on Information and System Security (TISSEC)*, 13(1):1–35, November 2009.
[26] L. Shundong, W. Daoshun, D. Yiqi, and L. Ping. Symmetric Cryptographic Solution to Yao's Millionaires' Problem and an Evaluation of Secure Multiparty Computations. *Information Sciences*, 178(1):244–255, 2008.
[27] Q. Xie. Privacy-Preserving Interest Matching for Mobile Social Networking. Master's thesis, University of Waterloo, 2010.