Implementing Privacy as Symmetry in Location-aware Systems

Anders Kofod-Petersen, Espen Klæboe, Jørgen Jervidalo, Kjetil Aaltvedt, Magnus Romnes, and Trond Martin Nyhus

Department of Computer and Information Science,
Norwegian University of Science and Technology,
7491 Trondheim, Norway
`anderpe@idi.ntnu.no`,{`espenkl`|`jervidal`|`kjetilue`|`romnes`|`trondmn`}`@stud.ntnu.no`

Abstract. Social network services on the internet are moving from a traditional web-based service into ubiquitous computing environments. With this migration these services will also benefit from the context-awares that ubiquitous computing offers. Applications that sense the environment and act proactively, requires an immaculate attention to users' privacy. The work presented here approaches privacy by employing the *principle of minimum asymmetry* to a mobile social network service. We demonstrate how this principle can be implemented on a simple location-aware application running on standard mobile telephones in a traditional GSM network.

1 Introduction

The number and popularity of digital social networks have been steadily increasing over the last few years. Privacy and trust are important issues in these social aware applications. The importance will increase tremendously when social aware applications are moved into mobile or pervasive applications. When applications assume responsibility from the user and act proactively, as pervasive applications do by definition, the control over what information is shared and to whom becomes paramount.

One important approach to maintain privacy and trust in pervasive applications is the principle of minimal asymmetry, which in short states that the ability to obtain information should be coupled with the sharing of information. The work presented here demonstrates how how this principle can be applied in a mobile social-aware application. The application implements location-awareness on standard mobile phones in running in a GSM network.

The rest of the paper is organised as follows: First, an overview of related work concerning privacy in ubiquitous computing is presented. This is followed by a description of the systems design and implementation. The paper ends with a summary and pointers to future work.

2 Related Work

Social systems that link people to people, and people to geographical places are referred to as P3 systems [1]. P3 systems can be divided into two difference categories: *people-centered* and *place-centered.* An example of a people-centered system might be one where a user has a contact list, where the contacts show up in different colours (green, yellow, red) depending on their proximity. Jones and Grandhi presented a survey executed at various places in Manhattan with more that 500 participants. Among other things, they discovered that 84% of the participants were willing to share their location data (anonymously) to get information about crowding and occupancy in public places. They concluded that a large population considered P3 systems to be sufficiently beneficial to disclose their position. The fact that this percentage of the population were willing to give away their position in exchange for a service that they considered beneficial is an important insight when modelling context-aware systems, and perhaps even more important when systems are to reason about what (contextual) information they can share and to whom.

With regard to privacy in context-aware systems, Langheinrich [2] describes why privacy is of particular importance in ubiquitous computing with four properties: *ubiquity*, computers are everywhere; *invisibility*, computers disappear from the scene; *sensing*, sensors are becoming more precise; and *memory amplification*, storing of large amount of (sensed) data.

Many of the privacy issues in context-aware systems are related to the issue of *mutual awareness.* One part of this problem is about *disembodiment* and *dissociation.* When we encounter people in the real world we can receive information in many ways, such as position, voice level, face expression and direction of gaze. In ubiquitous environments these communication channels are likely to be less effective. In real life people live by the intuitive principle that if you cannot see me, I cannot see you. Due to the potential large number of sensors in an ubiquitous environment, this is not always true. Users may not always know exactly what information they are conveying, in what form, if it is permanent, and to whom they are sending [3].

Jiang et al. [4] discusses the *principle of minimum asymmetry* when dealing with privacy issues in ubiquitous computing. This principle goes a long way towards handling the apparent asymmetric relationship between sender and receiver of information, as described by Bellotti et al. [3]. Jiang et at. argue that a privacy-aware system should minimise the asymmetry of information between data owners and data users. For an example, if a user does not wish to share his location he cannot expect others to share their location with him (regardless of their wish). The main principle is that [4, p. 7] (original emphasis):

> A privacy aware system should' minimize the asymmetry of information between **data owners** and **data collectors and data users**, by:
>
> – **Decreasing** the flow of information from data owners to data collectors and users

– **Increasing** the flow of information from data collectors and users back to data owners

When decreasing the flow of information from the data owner the user maintains a higher degree of control over the system. Increasing information flow from the data collector provides better feedback to the user. Examples of mechanisms that adhere to this principle are: *anonymising* or *pseudonymising*, which approaches privacy by allowing users to act in total anonymity or through a consistent avatar that cannot be coupled with a real person. Further, *plausible deniability* allows the user to plausible deny that he did not wish to interact with a particular person, at a given time [5]. Finally, *reciprocity* is the essential property for building trust and deep relationships. Sharing too little or too much information might have a negative effect on relations to one's peers. Balancing the amount on information flowing between peers are very important to maintain a balance in any relationship. Social systems often approaches this by for an example only allowing you to see the status of the people whom you allow to see your status. To receive better feedback from the system it might log all access to one's position data, notify the user when somebody requests a position, and give clear feedback on what information is stored [4].

Lederer et al. [6] argues that feedback and control is "the designer's opportunity to empower those processes (understanding and action), and they are the user's opportunity to practice them." The authors exemplify pitfalls in design of systems maintaining privacy using their personal experience. The pitfalls can be divided into two main groups: feedback and control. The feedback pitfalls are: *obscuring potential information flow*, where systems do not explicitly describe the possible disclosures it can make; and *obscuring actual information flow*, where a system might not explicitly make clear what information is actually disclosed. The control pitfalls are: *emphasising configuration over action*, where configuration overshadows the privacy management actually needed to adapt to a user's ordinary use of the system; *lacking coarse-grained control*, where a system offers too many choices and not just simple on/off choices; and *Inhibiting existing practice*, where a system forces some required practice onto a user, and does not adapt to the user's practice.

As aforementioned, digital social networks have recently emerged, and along with them several communication protocols and representational models. Among these are: XHTML Friends Network (XFN), Friend of a Friend (FOAF) and Extensible Messaging and Presence Protocol (XMPP) commonly known as Jabber.

XFN[1] is a decentralised markup language that captures social connectivity. This language can represent different forms of social connectivity, such as the level of friendship, professional relations, geographical proximity, family ties and romantic aspects. XFN maintains some privacy related limitations, such as the fact that many personal attributes cannot be assigned by others. Some examples of these include: gender, race and age. This limitation is founded in the fact that a user describes a *relation* to another person from that user's perspective, and

[1] http://gmpg.org/xfn/

not the friend. Since a “friendship” in XFN is directional relationship managed solely by the user who initiates a relation, the principle of minimal asymmetry seems hard to maintain.

FOAF[2] is like XFN a modelling language used to represent social relations. FOAF have chosen RDF, a more expressive language that XHTML chosen by XFN. FOAF is currently an immature language, which does not explicitly cover issues such as control of data within a community of trust, guaranty of facts and general privacy of data.

XMPP[3], or Jabber as it is commonly know, is a message protocol for instant messaging services [7,8]. The Jabber specification defines a XML streaming protocol that covers not only instant messaging but also issues such as presence. As the XML protocol is extensible, the core functionality such as authentication, privacy mechanisms and chatting are easily extensible.

3 Design and Implementation

Find Per Anton (hereby referred to as ”the application”) is a location-aware instant messaging application developed for mobile phones. It utilises the underlying TCP/IP-network capabilities (such as GRPS, UMTS and Wi-Fi) of the mobile phone and will supplement a similar application based on Wi-Fi technology [9]. The localisation service is obtained from the third party provider Geomatikk[4] and all transmissions between the server and the client uses the XMMP protocol.

The idea is to have the opportunity to sort your friends in groups, locate one friend or a whole group on the map and also be able to send messages to contacts or whole groups. The application provides feedback about the locations of the peers of a user by indicating their proximity using colours ranging from red to green and by showing their position on a map. An example of how the map service might look can be seen in Fig. 1.

Table 1. Presence states allowed by the XMPP standard.

Name:	**Description:**
offline	The entity or resource is not connected to the server.
chat	The entity or resource is actively interested in chatting.
away	The entity or resource is temporarily away.
dnd	The entity or resource is busy (dnd = ”Do Not Disturb”).
xa	The entity or resource is away for an extended period (xa = ”eXtended Away”).

[2] http://www.foaf-project.org/
[3] http://www.xmpp.org/
[4] http://www.geomatikk.no/

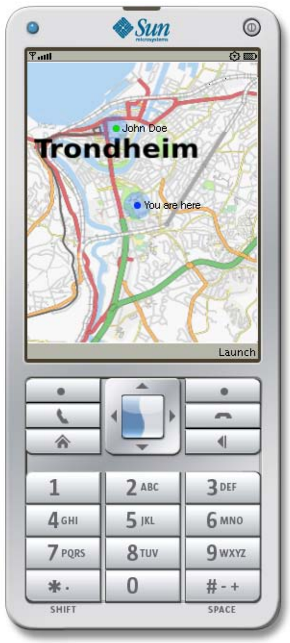

Fig. 1. An example of the map service running on a mobile device

As mentioned above the application design builds on the XMPP standards [7,8]. Unfortunately, XMPP does not enforce the principle of minimum assymmetry, which is a requirement for the application. For this reason, a deviation was made between the original specification and the application design by only supporting a subset of the XMPP subscription states. The subscription states supported by the original XMPP specification can be seen in Table 2. Every subscription state except *None* and *Both* was omitted from the application design because they contradict the principle of minimum asymmetry by allowing a user to receive information without releasing information himself. In the XMPP standard, adding a contact is refered to as *subscribing to a contact* and a contact list is refered to as a *roster list*, so we will use these terms from now on.

In order to establish a subscription, both users must agree to share *presence information* and *location information* with the other part. This requirement allows the application to enforce the principle of minimum asymmetry. The presence information contains the current status of the user, which may be *away*, *chat*, *dnd*, *xa* or *offline*. A detailed description of the presence status can be found in Table 1. The location information contains the longitude and latitude of a user, and will appear graphically as a position on a map. This way a user can physically locate the position of any of his contacts, in return for being discoverable himself. A user can temporarily make himself invisible to others, but he is then no longer entitled to receive any location information, until he

Table 2. Subscription states supported by the XMPP standard, described from the user's perspective.

Name:	**Description:**
"None"	Contact and user are not subscribed to each other, and neither has requested a subscription from the other.
"None + Pending Out"	Contact and user are not subscribed to each other, and user has sent contact a subscription request but contact has not replied yet.
"None + Pending In"	Contact and user are not subscribed to each other, and contact has sent user a subscription request but user has not replied yet.
"None + Pending Out/In"	Contact and user are not subscribed to each other, contact has sent user a subscription request but user has not replied yet, and user has sent contact a subscription request but contact has not replied yet.
"To"	User is subscribed to contact (one-way).
"To + Pending In"	User is subscribed to contact, and contact has sent user a subscription request but user has not replied yet.
"From"	Contact is subscribed to user (one-way).
"From + Pending Out"	Contact is subscribed to user, and user has sent contact a subscription request but contact has not replied yet.
"Both"	User and contact are subscribed to each other (two-way).

makes himself discoverable again, as discussed by Jiang et. al [4]. When he makes himself invisible, he will appear as offline to his contacts, just as if he had turned off his mobile phone.

When a user cancels a subscription, the contact is not notified. The contact is removed from the user's roster list, but the user is only shown as offline on the contact's roster list, thus preserving minimum asymmetry and plausible deniability.

In Listing 1.1, we show a simplified pseudo code example of how a user adds a subscriber (contact) to his roster list in the application. The client sends a request for a new subscription to the server, using the method `newSubscription()`. The server checks the current subscription state for the subscriber. In an initial state, there will not exist any relations between the user and the contact, so the server will store the new subscription state (*None*) and send a subscription request to the contact. If the contact accepts the subscription request, his client will send a subscription request to the server, again using the same `newSubscription()`-method. This time, when the server receives the request, there already exists a relation between the user and the contact, so the server sets the subscription states to *Both* for the user and the contact, adds them to the respective roster lists and pushes the new roster lists to both clients. The symmetrical subscription state *Both* is the *only* state where positioning, presence notifications or messaging is allowed.

Listing 1.1. Implementation Pseudo Code (Add a Friend)

```
// [CLIENT] User 1 requests to add User 2 as a friend:
function addFriend(newcontact)
{
  // Send a request for a new subscription to the server:
  server.newSubscription(currentuser, newcontact)
}

// [SERVER] Receiving notification about new subscription
function newSubscription(user, contact)
{
  if contact.hasAccepted(user)
    // Sets the subscription state for both parties and
    // add 'user' to 'contact's roster list:
    setSubscriptionState(contact, user, BOTH)
    setSubscriptionState(user, contact, BOTH)
    addToRoster(contact, user)

  else if contact.hasDenied(user)
    // Don't make the user ask the contact for permission
    // any more:
    setAskForAcceptance(user, contact, FALSE)

  else
    // As long as the contact is undeceided or has refused,
    // we use NONE. However, user has contact on his roster
    // list
    setSubscriptionState(user, contact, NONE)
    addToRoster(user, contact)
    // Ask User 2 (contact) to accept symmetrical subscription
    // to User 1
    askForAcceptance(contact)
}

// [CLIENT] User 2 (currentuser) receives subscription
// request from User 1 (contact)
function onSubscriptionRequestEvent(contact)
{
  // Show GUI asking for acceptance
  if accepted
    // User 2 accepted a symmetrical subscription to User 1
    server.newSubscription(currentuser, contact)
}

// [CLIENT] User 1 receives updated
function onRosterUpdateEvent()
{
  redrawRoster()
}
```

When a user wants to remove a subscriber from his roster list, we must not only enforce the principle of minimum asymmetry, but also retain privacy for the user. In other words, the removal has to be done in a discrete way. Pseudo code showing how this is handled in the system, can be seen in Listing 1.2. The user's client calls the `deleteSubscription()`-method on the server, which removes the specified subscriber from the user's roster list. Note that the user **is not** removed from the subscriber's roster list. Instead the removed contact's subscription state to the user is set to *None* on the server. This will result in the user to always appear offline to the contact.

By applying these methods to the application, we achieve minimum asymmetry both with respect to presence information and location information.

Listing 1.2. Implementation Pseudo Code (Remove a Friend)

```
// [CLIENT]: User 1 wants to delete User 2 (contact)
function deleteSubscription(contact)
{
  server.deleteSubscription(currentuser, contact)
}

// [SERVER]: User 1 wants to delete User 2 (contact)
function deleteSubscription(user, contact)
{
  // Remove contact from user's roster:
  removeFromRoster(user, contact)

  // Set user's subscription state to NONE in contact's
  // roster. Contact will see user as offline.
  setSubscriptionState(contact, user, NONE)

  // Send a new roster to user without contact in it.
  sendUpdatedRoster(user)
}
```

4 Summary and Further work

The work presented here has demonstrated how the *principle of minimum asymmetry* can be applied as a means of maintaining privacy in a mobile social application. The service uses the XMPP protocol to exchange messages between different users. However, the XMPP standard does not conform to the principle of minimum asymmetry. By using a subset of the standard, we show how we can enforce mutual subscriptions and thus minimum asymmetry.

A similar system has been developed for the Wi-Fi environment in Wireless Trondheim [9]. This system has primarily been developed to investigate any behavioural changes in the users when they *know the location of their peers* and *they know that their peers know their location*[5]. We expect to conduct similar experiments on a larger scale, using the implemented system described in this paper. The experiment will be conducted using the GSM mobile network in Norway.

Acknowledgements

Parts of this work has been supported by Accenture Innovation Lab Norway. We would like to extend our thanks to Per Anton Gransæther for his assistance.

References

1. Jones, Q., Grandhi, S.A.: P3 systems: Putting the place back into social networks. IEEE Internet Computing **9** (2005) 38–46
2. Langheinrich, M.: Privacy by design – principles of privacy-aware ubiquitous systems. In Abowd, G.D., Brumitt, B., Shafer, S.A., eds.: Proceedings of the Third International Conference on Ubiquitous Computing (UbiComp 2001). Number 2201 in Lecture Notes in Computer Science, Springer Verlag (2001) 273–291
3. Bellotti, V., Sellen, A.: Design for privacy in ubiquitous environments. In Michelis, G.D., Simone, C., Schmidt, K., eds.: Proceeding of the Third European Conference on Computer-Supported Cooperative Work (ECSCW '93), Kluwer Academic Publishers (1993) 77–92
4. Jiang, X., Hong, J.I., Landay, J.A.: Approximate information flows: Socially-based modeling of privacy in ubiquitous computing. In Borriello, G., Holmquist, L.E., eds.: Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp 2002). Volume 2498 of Lecture Notes in Computer Science., Springer Verlag (2002) 176–193
5. Raento, M., Oulasvirta, A.: Privacy management for social awareness applications. In Floréen, P., Lindén, G., Niklander, T., Raatikaine, K., eds.: Workshop on Context Awareness for Proactive Systems (CAPS 2005), HIIY Publications (2005) 105–114
6. Lederer, S., Hong, I., Dey, K., Landay, A.: Personal privacy through understanding and action: five pitfalls for designers. Personal and Ubiquitous Computing **8** (2004) 440–454
7. Saint-Andre, P.: Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920 (2004)
8. Saint-Andre, P.: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. RFC 3921 (2004)
9. Andresen, S., Krogstie, J., Jelle, T.: Lab and research activities in wireless trondheim. In: Proceedings of IEEE International Symposium on Wireless Communication Systems, IEEE Computer Society (2007) 385–389

[5] This research is currently being prepared for publication