Privacy Issues in an Address Book

Dong-Young Lee and YounSung Chu
LG Electronics
{dongyoung.lee, ys.chu}@lge.com

Introduction

Privacy is an important issue on a mobile phone, which is a personal device by nature. Among features of a mobile phone, address book is of particular importance since it is a database of personal information, not only of the mobile phone user but also of his/her contacts.

Evolution of Address Books

In a traditional address book, contact information is stored in a mobile phone and consumed only by its user/owner. Sending contact information using vCard via email or MMS might be the only exception. Since information is consumed only by its owner and is not shared, there was not much privacy issue in the address book application itself.

Traditional address books have been evolved to network address book services, in which contact information is stored at a server on the network side. However, the information is still not (supposed to be) shared to other users or any third party. Thus in terms of privacy, a network address book service is equivalent to a traditional address book stored in a phone, as long as only the user/owner can access the network address book.

Converged Address Book (CAB) is a service enabler for future address book service being standardized by Open Mobile Alliance (OMA). An innovative aspect of CAB is that it is designed for sharing contact information as primary features, which would have privacy implication. In addition to an address book, which contains contact information of *other* users, each user keeps a "Personal Contact Card (PCC)", which contains the user's *own* contact information. Information in a PCC may be shared to other users, and can even be made public. Features of CAB include:

- Contact Subscription: A user is able to subscribe to another user's PCC. When a PCC is updated that a user has subscribed to, the change is propagated to the user and the user's address book can be automatically updated.
- Contact Share: A user is able to send information in his/her PCC or address book to other users.
- Access Control: A user is able to specify access permission of other users for his/her contact information. That is, it is possible to allow or disallow other users to access his/her address book and to subscribe to his/her PCC. The user can also control the extent of information that is exposed by a search. The access control mechanism is fine grained such that access permission of a user for an operation on a field may be specified.

- Contact View: A user is able to organize information in his/her PCC into contact views, each of which contains a subset of information in the PCC. Contact views may be used to implement "multiple persona", which means that a user may have contact views that have different names and contact information.
- Import: A user is able to import contact information from other address book systems.

New Challenges

Although sharing information is supplemented by access control, these new features involve new privacy challenges that are not addressed by a plain access control mechanism based on the requestor identity.

- A user with multiple personae may not want others to know the different personae are actually the same person. Note that this problem is not specific to a system that specifically supports multiple persona. Since a person may use different services, the person has a "persona" for each service he/she uses. For example, a person who is "John Smith" in Facebook and "FluffyBunny" in Second Life may not want to disclose that the two identities are of the same person. On the other hand, when users import contact information from other address book systems, they would like contact information of the same person from different address books to be merged into a single entry.
- A user may have a public contact view, which would perhaps include basic contact information about the user. A directory of such public contact view can easily be abused. For example, an attacker may try to collect email addresses from a public directory. Furthermore, exposing contact views is more dangerous than exposing a list of a single type of contact information. For example, a cell phone number by itself is barely more valuable than a random number, but the number along with the user's name and email address is very valuable for an attacker and thus very dangerous to expose. Such a risk would prevent most users from putting significant information in their public contact view, and the directory would end up with no more than a list of boilerplates, or a very short list of meaningful information. On the other hand, however, users may still would like to have public contact views to help innocent others to be able to find their contact information.

The above two challenges in fact boils down to the same problem: How to authorize a search? For the first challenge, matching entries for the same person in address book system A and B is essentially for system A to search for his/her identifier in system B based on common information, or vice versa. For the second challenge, what a user really would like may be not really to expose his/her personal information to everybody, but to allow cannot-be-specified-in-advance others to look up the information. Then how could we specify the "cannot-be-specified-in-advance" others?

Authorization Based on Prior Knowledge

To solve the authorization problem, we observe two things:

- In case of matching, what is really necessary is the identifier of the target entry in the searched system. In an extreme case, we may not want allow any more information except for the identifier to be exposed.
- In case of public directory, an innocent search requester usually already knows the target user, whereas malicious requester (attacker) tries to collect personal information.

From the above observations, we can think of an approach to solve the authorization problem: The search requester proves that he/she already knows the target user by providing enough prior information about the target user. The prior information should also be specific enough to be able to uniquely identify the target user. Globally-unique identifiers, e.g., an email address, a Facebook id, a cell phone number with country and area codes would be ideal to specify the target user. Perhaps combination of two or more of such identifiers would be enough to prove that the requester already knows the target user.

A problem in practice is how the requester knows how much and what information should be provided to get authorized. A straightforward solution is including the authorization rule in the response message. An authorization rule in XML would be convenient to be embedded in a response message.

Protecting Requesting Side

One may argue that with the prior-knowledge-based authorization approach described above, the requestor gets a risk of information leak. Note that the purpose of prior knowledge included in a search request is only to confirm that the requestor already has the knowledge, rather than to transfer the knowledge. A hash is an ideal solution for such a purpose. That is, in case the target system is not trusted, the request may provide hash values of the prior knowledge in the search request rather than providing the information itself. (We do not discuss further details here, e.g., use of a challenge-response mechanism or a nonce for protection against a replay attack.)

Hash functions can also be used to improve efficiency. In an environment where several independent systems form a federation and frequently query one another, a hash function, or more desirably a Bloom filter, can be used to filter out unnecessary queries.

Conclusions

Anonymity is an essential part of privacy. A person would not be able to be anonymous if the same

identifier is used in every system and service that he/she uses. Therefore using different identifiers in different systems should be the first step for anonymity.

Once the first step is satisfied, the next step would be properly protecting mapping between identifiers of the same person. We see this issue in practice, e.g., when we try to import Facebook buddy list and Outlook address book into a unified address book on a mobile phone. A database that contains people's Facebook id, email addresses, mobile phone numbers, etc. would enable a very convenient user experience, as much as such a large collection of personal information is dangerous.

Although we know that convenience and danger are two sides of the same thing, convenience often wins danger with a justification of user's authorization. In such cases, what needs to be ensured is a mechanism that can precisely express what a user would like to authorize.