An Architecture for Privacy-Sensitive Ubiquitous Computing

Jason I. Hong
Group for User Interface Research
Computer Science Division
University of California at Berkeley
Berkeley, CA, 94720-1776 USA
jasonh@cs.berkeley.edu

James A. Landay
DUB Group
Computer Science and Engineering
University of Washington
Seattle, WA 98105-4615 USA
landay@cs.washington.edu

ABSTRACT

Privacy is the most often-cited criticism of ubiquitous computing, and may be the greatest barrier to its long-term success. However, developers currently have little support in designing software architectures and in creating interactions that are effective in helping end-users manage their privacy. To address this problem, we present Confab, a toolkit for facilitating the development of privacy-sensitive ubiquitous computing applications. The requirements for Confab were gathered through an analysis of privacy needs for both end-users and application developers. Confab provides basic support for building ubiquitous computing applications, providing a framework as well as several customizable privacy mechanisms. Confab also comes with extensions for managing location privacy. Combined, these features allow application developers and end-users to support a spectrum of trust levels and privacy needs.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques – *Software libraries.* D.2.11 [**Software Engineering**]: Software Architectures – *Domain-specific architectures.* H.1.2 [**Models and Principles**]: User/Machine Systems – *Human factors.* H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *User-centered design.* K.4.1 [**Computers and Society**]: Public Policy Issues – *Privacy.*

General Terms

Design, Security, Human Factors

Keywords

Ubiquitous computing, privacy, toolkit, Confab, location

1. INTRODUCTION

Westin defined information privacy as "the claim of individuals, groups or institutions to determine for themselves when, how, and to what extent information about them is communicated to others" [70]. While many people believe that ubiquitous computing holds great promise, privacy is easily its most often-cited criticism. Numerous interviews (e.g. [8, 35, 43]), essays (e.g. [21, 67, 69]), books (e.g. [11, 28]), and negative media coverage (e.g. [64, 71]) have described people's concerns about the strong potential for abuse, general unease over a potential lack of control, and desire for privacy-sensitive systems. These concerns suggest that privacy may be the greatest barrier to the long-term success of ubiquitous computing.

The large majority of previous work on privacy has tended to focus on providing anonymity or on keeping personal information and messages secret from hackers, governments, and faceless corporations. While anonymity and secrecy are clearly important, they only address a relatively narrow aspect of privacy and do not cover the many situations in everyday life where people *do* want to share information with others. For example, one could imagine sharing one's location information with friends to facilitate micro-coordination of arrivals at a meeting place, or sharing simple notions of activity to convey a sense of presence to co-workers and friends. It is important to note here that the parties that are receiving such information already know one's identity, are not adversaries in the traditional sense, and that the privacy risks may be as simple as wanting to avoid undesired social obligations or potentially embarrassing situations.

The point is that, rather than being a single monolithic concept, privacy is a fluid and malleable notion with a range of trust levels and needs. Our focus here is in empowering people with choice and informed consent, so that they can share the right information, with the right people and services, in the right situations. As Weiser noted, "The problem, while often couched in terms of privacy, is really one of control. If the computational system is invisible as well as extensive, it becomes hard to know what is controlling what, what is connected to what, where information is flowing, how it is being used…and what are the consequences of any given action" [69].

However, the problem is that it is still difficult to design and implement privacy-sensitive ubicomp applications. Previous work, such as the PARCTab system [61], the Context Toolkit [20], and iROS [42], provide support for building ubicomp applications, but do not provide features for managing privacy. Consequently, system developers have little guidance or programming support in creating architectures and user interfaces that are effective in helping end-users manage their privacy. The result is that privacy is done in an ad hoc manner and often as an afterthought, if at all, leading to applications that end-users may ultimately reject because they are uncomfortable using them or find them intrusive.

Based on an analysis we performed of end-user privacy needs and developer privacy needs, we have developed Confab, a toolkit

MobiSYS'04, June 6–9, 2004, Boston, Massachusetts, USA.

facilitating the construction of privacy-sensitive ubicomp applications. Confab is tailored for context-aware computing [62], an aspect of ubiquitous computing in which sensors and other data sources are leveraged to provide computing systems with an increased awareness of a user's physical and social environment. From a software architecture perspective, Confab provides a framework and an extendable suite of privacy mechanisms that allow developers and end-users to support a spectrum of trust levels and privacy needs. This framework is designed such that personal information is captured, stored, and processed on the end-user's computer as much as possible. Afterwards, end-users can choose what information to share with others. This approach provides end-users with greater control over disclosures than previous approaches. This constraint can also be relaxed in situations with greater trust or fewer privacy concerns, such that capture, storage, and processing are done on other computers.

From an end-user perspective, Confab facilitates the creation of three basic interaction patterns for privacy-sensitive applications [33]: *optimistic*, where an application shares personal information and detects abuses by default; *pessimistic*, where it is more important for an application to prevent abuses; and *mixed-initiative*, where decisions to share information are made interactively by end-users.

It should be noted that Confab is not intended to provide perfect privacy, if there is even such a thing. There are many social and organizational issues that simply cannot be managed by technological means alone. Ultimately, privacy will have to be managed through a combination of technology, legislation, corporate policy, and social norms [50]. What Confab does provide is a more solid technical foundation for privacy-sensitive ubiquitous computing than previous approaches, making it easier for developers to build privacy-sensitive applications for an intended community of users and for companies to offer their services while minimizing the risk to people's privacy. As an analogy, a web design tool can be used to create good as well as bad web sites, but a useful tool will be oriented to make it easier to create good ones.

The rest of this paper is organized as follows. First, we examine the requirements for a toolkit for privacy-sensitive applications. We continue with a description of Confab's architecture and how it supports those requirements, including some specific mechanisms for managing location privacy built within Confab's framework. Then, we describe our evaluation with three applications we have built on top of Confab. We wrap up with a comparison to related work, a brief discussion of future work, and conclusions.

2. SYSTEM REQUIREMENTS

The primary metric of success for any toolkit is if it can be used to create a useful and non-trivial subset of the full design space of applications in a manner that is faster, is higher quality, or has more useful features than without it. In this section, we explore the targeted design space, looking at the requirements we gathered by analyzing end-user and application developer needs.

2.1 End-User Needs

The end-user needs for Confab were gathered through scenario-based interviews on location-enhanced applications we performed with twenty people of various ages and computer expertise; extended analysis of freeform comments we performed on a previous survey of 130 people on ubicomp privacy preferences [49]; analysis of research papers [8, 14, 32, 35, 37, 38, 43, 49] on hypothetical and actual usage of emerging ubicomp systems where privacy was an issue, as well as postings on a message board [1] by nurses working in hospitals using locator systems; analysis of several proposed and existing privacy protection laws [2, 22, 27]; and analysis of several different design guidelines for privacy-sensitive systems [5, 9, 55], in particular the fair information practices [48, 70] and asymmetric information flows [41]. A full discussion of the results of our analysis are beyond the scope of this paper. Instead, we provide a summary of six major themes, as shown in Table 1.

End-user requirements
• Clear value proposition
• Simple and appropriate control and feedback
• Plausible deniability
• Limited retention of data
• Decentralized control
• Special exceptions for emergencies

Table 1. Summary of end-user requirements.

First, applications need an upfront value proposition that makes it clear what benefits are offered and what personal information is needed to offer those benefits. This need was most pronounced in the early active badge systems [68] and in the nurse message board [1], where a lack of a clear value proposition led to resentment and sometimes outright rejection of a system.

Second, people want simple control over and feedback about who can see what information about them [1, 14, 35, 37, 38]. For example, the PARCTab system provided no feedback about what information was being revealed to others [38, 61]. There were serious concerns that a co-worker or boss could monitor a user's location by making repeated queries about the user's location without that user knowing. Previous research has focused on flexible, yet complex access control mechanisms; however, our surveys and interviews suggest that in many cases, simple access control and basic notifications are sufficient. Access control can be used to select who can view personal information, with notifications providing social visibility to prevent abuses. For example, Alice is less likely to repeatedly query Bob's location if she knows that Bob can see each of her requests.

There are also concerns about continuous versus discrete flows of information. Many of our interviewees said they would be comfortable with co-workers getting snapshots of their current location, but would be less comfortable continuously sharing their location information, as that could be used to monitor them.

Third, people expressed a strong desire for plausible deniability. Our survey and interviews, as well previous work on ubicomp in the home [37], have suggested a social need to avoid potentially embarrassing situations, undesired intrusions, and unwanted social obligations. For example, it is not uncommon for an individual to answer with a white lie when asked on the phone what they are doing. Cell phones are a good example of a system that provides plausible deniability. If a person does not answer a call, it could be for technical reasons—such as being outside of a cell, not having the phone with them, or that the phone is off—or for social reasons, such as being busy or not wanting to talk to the caller right now. By default, it does “the right thing” without the end-user having to take any special action.

Fourth, some of our interviewees were concerned over long-term retention of personal information, as it opens up the possibility for extensive data mining. Limited data retention is also an issue explicitly espoused by data protection laws [2, 22, 27].

Fifth, people are concerned about systems that centralize data [8, 38]. While there are many advantages to centralized architectures, it also means that sensitive data is stored on a computer that end-users have little practical control over. For example, while a visible effort was made to create written privacy policies about how location information was used in the PARCTab system [61], users still had the perception that if the research team or upper-level managers wanted to examine the data, there was little they could do about it [38]. Similar debates have emerged over the deployment of E911 in the United States.

Sixth, people expressed the desire for special exceptions for emergencies. In crisis situations, safety far outweighs privacy needs. This sentiment was universal across all of our interviewees. Our interviews also noted that E911 made sense if it transmitted location information only when making the call, and not at any other time. Trusted proxies are sometimes used to handle these kinds of situations. For example, MedicAlert [3] is a paid service that stores personal medical records and forwards it to emergency responders in the case of medical emergencies.

2.2 Application Developer Needs

The application developer needs for Confab were gathered by identifying privacy functions common in several networked as well as ubicomp applications. We examined research prototypes and emerging commercial applications, limiting the scope to what we call *personal ubiquitous computing*, systems where data starts with the end-user and can optionally be disclosed to others in a limited manner. We also chose to focus more on location than on other forms of contextual information, since a sizeable number of this type of application is emerging in the market, and thus has a clearer path to widespread use. We were also influenced by the Geopriv working group’s requirements for location privacy [18] and our previous work on asymmetric information flows [41].

The genres of applications we have examined include messaging systems, such as cell phones, instant messenger, SMS, and messaging within [52] and between homes [37]; guides for exploration and navigation [4, 54]; finders for finding people, places, or things [7, 31]; group awareness displays [20, 31]; augmented-reality games [25, 29]; contextual tagging and retrieval, including personal memory aids [12, 46, 59], associating topical information with places [13, 24, 56, 62]; situational real-time information (such as local weather or traffic); and enhanced safety for individuals and emergency responders [26, 51].

From a systems standpoint, there are several basic features that need to be supported, including acquiring context data from a variety of sources, refining and storing that context data, and retrieving and using context data. This last issue, retrieving and using, can be done either through push transactions (e.g., you send your location in an E911 call) or pull transactions (e.g., a friend requests your location). For each of these types, there is also a need for continuous sharing, where personal data is constantly forwarded to another party (e.g., continuously sharing health information with your doctor), as well as for discrete disclosures that happen intermittently or one time only. These are basic features that are mostly supported by other systems aiding the development of ubicomp applications (e.g. [20, 61]).

Application Developer requirements
• Support for optimistic, pessimistic, and mixed-initiative applications
• Tagging of personal information
• Mechanisms to control the access, flow, and retention of personal information (quantity)
• Mechanisms to control the precision of personal information disclosed (quality)
• Logging

Table 2. Summary of developer requirements.

From a privacy standpoint, we have identified six common features that need to be supported (see Table 2). The first is support for the three basic interaction patterns for privacy-sensitive applications: pessimistic, optimistic, and mixed-initiative [33]. In *pessimistic* applications, end-users set up preferences beforehand, placing strict requirements on when personal information can flow to others. In contrast, *optimistic* applications [57] are designed to allow greater access to personal information but make it easier to detect abuses after the fact with logs and notifications. For example, AT&T mMode’s Find Friends [7] provides a notification each time a friend requests your location. Optimistic access control is useful in cases where openness and availability are more important than complete protection. Optimistic access control is also easier to use, since it is difficult for people to predict all of the possible usage scenarios they might find themselves in, and thus all of the necessary permissions. In *mixed-initiative* control, end-users are interrupted when someone requests their personal information and must make a decision then and there. An example is choosing whether or not to answer a phone call given the identity of the caller.

The second is support for tagging personal information as it flows to others, as described by Geopriv [18] and by Korba and Kenny [45]. Personal information can be marked with preferences about, for example, whether it should be forwarded to others or how long it should be retained. These tags can be thought of as applying Digital Rights Management for privacy purposes, and can be used as a fingerprint to help with tracking and auditing as well.

The third privacy need is mechanisms for controlling the access, flow, and retention of personal information, i.e. the quantity of personal information disclosed to others. These include restrictions based on identity, location (e.g., only allow inquirers in the same building as me to see my location), and time (e.g., co-workers can see my location between 9AM and 5PM), as well as invisible mode, a common feature in instant messenger clients where no information is disclosed.

The fourth necessary feature is granular control over the precision of disclosures, i.e. the quality of disclosures. One could choose to disclose one’s location as “123 Main Street” or “Atlanta”, or one’s activity as “writing a paper” or “busy” depending on who is requesting the information and on the current situation.

The fifth common privacy feature is logs, both for clients and servers. On the client side, logs that are summarized in a compact form make it easier for end-users to understand who is accessing what data. On the server side, logs make it easier for service providers to audit their activities to ensure that they are handling their customers' personal information properly. On both sides, logs also make it possible to apply machine learning techniques to detect unusual access patterns that might indicate abuses of someone's personal information.

2.3 Summary of Requirements

We have organized the end-user and application developer requirements into four high-level requirements below.

- A decentralized architecture, where as much personal information about an end-user is captured, stored, and processed on local devices owned by that end-user
- A range of mechanisms for control and feedback by end-users over the access, flow, and retention of personal information, to support the development of pessimistic, optimistic, and mixed-initiative applications.
- A level of plausible deniability built in
- Special exceptions for emergencies

It is important to note here that the requirements presented above are intended to support a range of privacy policies rather than all being used in a single application. Different communities have different trust relationships. There is a spectrum of privacy needs, and ubiquitous computing applications should be tailored to those needs [41].

3. CONFAB SYSTEM ARCHITECTURE

Confab provides a framework for ubiquitous computing applications, where personal information is captured, stored, and processed on the end-user's computer as much as possible. This gives end-users a greater amount of control and choice than previous systems over what personal information is disclosed to others. In this section, we describe this framework, the built-in privacy mechanisms for Confab, as well as specific extensions for location privacy built within this framework. We describe these features with respect to the Confab's data model and programming model.

An important issue to address here is whether this kind of architecture is feasible, especially the capture of personal information in a privacy-sensitive manner. We believe that there will be a useful and non-trivial subset of ubicomp applications built along these lines, for two reasons. First, over the past few years, the research community has been moving from centralized location-tracking architectures (e.g., [68]) to decentralized location-support ones (e.g., [58, 63]) for reasons of scalability and privacy. We believe that future research will continue this trend in providing privacy protection in the physical sensor layer for other forms of personal contextual information. Second, there is already a large market for personal items in which sensors can be cheaply embedded, for example PDAs, home security systems, and cars. Although Confab could be used in cases where data is initially captured by others (e.g., smart rooms or surveillance cameras), we do not explicitly address those cases.

3.1 Usage Scenario

In this section, we describe two scenarios to help illustrate what kinds of applications we want to support and roughly how they would work within Confab.

Scenario 1 – Find Friend

Alice's workplace has set up a new server that employees can use to share their location information with one another. Employees can choose to share their location information by uploading updates to the server at the level they desire, for example at the room level, at the floor level, or just "in" or "out". To help allay privacy concerns, the server is also set up to provide notifications to a person whenever their location is queried, and to accept queries only if the requestor is physically in the same building.

Scenario 2 – Mobile Tour Guide

Alice is visiting Boston for the first time and wants to know more about the local area. She already owns a location-enabled device, so all she needs to do is find a service that offers an interactive location-enhanced tour guide and link her device to it. She searches online and finds a service named Bob that offers such tour guides for a number of major cities. She decides to download it and try it out.

When starting the application, Alice discovers that Bob offers three levels of service. If Alice chooses to share her location at the *city level*, Bob can tell her how long the lines are at major venues such as museums, and what calendar events there are. If she shares her location at the *neighborhood level*, Bob can also tell her what interesting shops there are and nearby points of interest. If she shares it at the *street level*, Bob can offer Alice all of the features described above, as well as real-time maps and a route finder that can help her navigate. The application also states that Bob will retain her location data for up to 3 months, and at the neighborhood level sends updates of her location to Bob every 10 minutes when the application is running.

Since this is her first time using the service, and since she has not heard of Bob before, Alice decides to share her location information at the neighborhood level.

3.2 Confab's Data Model

Confab's data model is used to represent contextual information, such as one's location or activity. People, places, things, and services (entities) are assigned *infospaces*, network-addressable

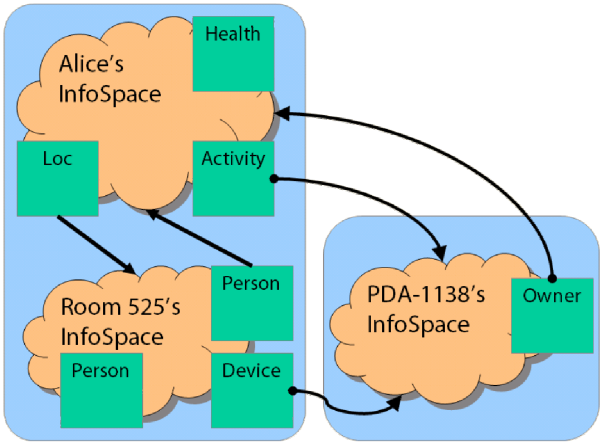


Figure 1. An infospace (represented by clouds) contains contextual data about a person, place, or thing. Infospaces contain tuples (squares) that describe individual pieces of contextual data, for example Alice's location or PDA-1138's owner. Infospaces are contained by Infospace servers (rounded rectangles).

logical storage units that store context data about those entities (see Figure 1). For example, a person's infospace might have static information, such as their name and email address, as well as dynamic information, such as their location and activity.

Sources of context data, such as sensors, can populate infospaces to make their data available for use and retrieval. Applications retrieve and manipulate infospace data to accomplish context-aware tasks. Infospaces also provide an abstraction with which to model and control access to context data about an entity. For example, individuals can specify privacy preferences for how their infospace handles access control and flow (described in greater detail below).

Infospaces are managed by *infospace servers*, which can be either distributed across a network or managed centrally, analogous to how a person could choose to have their personal web site hosted on their home machine or by an ISP. Here, we focus on the case where infospaces represent contextual information about individuals and are hosted on devices owned by those individuals.

The basic unit of storage in an infospace is the *context tuple*. Tuples are used to represent intrinsic context, that is an attribute about an entity (e.g., a person's age), as well as extrinsic context, which is a relationship between two entities (e.g., a person is in a room). Tuples are also used to represent static pieces of contextual information (e.g., an email address), as well as dynamic contextual information (e.g., a person's location). These different kinds of contextual information are summarized in Table 3.

Attributes of interest common to all tuples are *datatype*, a textual name describing the relationship of a tuple to the containing infospace's entity (for example, location or activity); *dataformat*, a string that describes the meaning of the data (for example, temperature could be Farenheit or Celsius); an optional *entity-link* denoting the address of an infospace for an entity described by the tuple; and one or more *values*, each identified by name (see Figure 2 for an example). Infospaces can store tuples containing arbitrary data, many of which may describe other entities related to the original infospace. Such tuples' entity-link attributes refer to the infospace of the other entity. For instance, the infospace for a specific room may contain numerous tuples of type 'occupant', each with values denoting a name and email of an occupant of the room and an entity-link referring to the infospace that hold tuples on behalf of that occupant.

Each tuple can also optionally have a *privacy tag* which describes hints provided by the end-user on how that tuple should be used when it flows to a computer outside of the end-user's direct control. The current implementation of privacy tags provides hints on when a tuple should be deleted, to help enforce limited data retention. End-users can have their tuples tagged with a *TimeToLive*, which specifies how long data should be retained before being deleted; *MaxNumSightings*, which specifies the maximum number of previous values that should be retained (for example, a value of 5 means only retain the last five places I was at); *Notify*, which specifies an address to send notifications of second use to; and *GarbageCollect*, which specifies additional hints on when the data should be deleted, for example, when the current holder of the tuple has left the area.

By default, when a tuple of any datatype is requested, its value is "UNKNOWN", regardless of whether it actually exists or not. Requests can see correct tuple values only if they have been granted access. This approach provides some level of plausible deniability, as a datatype might be unknown due to technical failures, lack of actual data, restricted access, or because the person is in invisible mode.

	Intrinsic	Extrinsic
Static	Name, age, email address	A room is part of a building
Dynamic	Activity, temperature	A person is in a specific room

Table 3. Confab supports different kinds of context data. Static context data does not change or changes very slowly, whereas dynamic context data changes often. Intrinsic context data represents information about that entity itself, whereas extrinsic context data represents information about an entity in relationship to another entity.

```
<ContextTuple dataformat="edu.school.building"
               datatype="location"
            description="location of an entity"
            entity-link="http://myhost.com/~jdoe"
            entity-name="John Doe"
      timestamp-created="2003.Feb.13 16:06 PST">

   <Values>
      <Value value="523" />
   </Values>

   <Sources>
      <Source datatype="location"
                  link="http://localhost/map.jsp"
                source="Location Simulator"
             timestamp="2003.Feb.13 16:06 PST"
                 value="523" />
   </Sources>

   <PrivacyTags>
      <Notify value="mailto:addr@mail.net" />
      <TimeToLive value="1 day" />
      <MaxNumSightings value="5" />
      <GarbageCollect>
         <Where requestor-location=
                   "not edu.school.building" />
      </GarbageCollect>
   </PrivacyTags>

</ContextTuple>
```

Figure 2. An example tuple. Tuples contain metadata describing the tuple (e.g., dataformat and datatype), one or more values, one or more sources describing the history of the data and how it was transformed, and an optional privacy tag that describes an end-user's privacy preferences. In this example, the privacy tag specifies a notification address, a maximum time to live, the maximum number of past values that should be retained, and an additional request to delete the data if the requestor is not in the specified location.

Infospace servers, infospaces, and context tuples are currently implemented using standard web technologies. Infospace servers are built on top of web servers, simplifying deployment and providing a clear mental model for programmers and end-users. Individual infospaces are named via URLs, and can be thought of as web-based tuplespaces with specialized constraints. Context tuples are currently represented as data-centric XML documents. That is, context tuples consist only of XML tags and XML attributes, with no text between tags.

3.3 Confab's Programming Model

From a high-level perspective, Confab is a hybrid blackboard and dataflow architecture. Personal information is stored in infospaces that are running in computers owned by end-users, with data flowing between these computers in a controlled fashion. In this section, we describe how developers can make use of three different pieces of functionality—operators, service descriptions, and active properties—to build applications.

Methods and Operators

Infospaces support two general kinds of methods, *in* and *out*. In-methods affect what data is stored within an infospace, and include add and remove. Out-methods govern any data leaving an infospace, and include query, subscribe, unsubscribe, and notify.

Each infospace also contains *operators* for manipulating tuples. Operators are chainable pieces of code that can be added to an existing infospace to extend and customize it to what is needed without having to modify the main body of code. Confab supports three different kinds of operators: in, out, and on. *In-operators* are run on all tuples coming in through in-methods. An example in-operator is one that checks the infospace's access control policies to make sure that this is a tuple that is allowed to be added. *Out-operators* are run on all tuples going out through out-methods. An example out-operator is one that blocks all outgoing tuples if the user is in invisible mode. *On-operators* are operators that run periodically, such as garbage collection. Table 4 shows a full list of operators provided in Confab by default.

Operator Type	Description
In	Enforce access policies Enforce privacy tags Notify on incoming data
Out	Enforce access policies Enforce privacy tags Notify on outgoing data Invisible mode Add privacy tag Interactive
On	Garbage collector Periodic report Coalesce

Table 4. Confab provides several built-in operators. Operators can be added or removed to customize what personal information a tuple contains and how it flows to others.

The two Enforce Access Policies operators (in- and out-) let end-users specify access policies for their infospace. Several different conditions can be specified for authorization, including who is requesting the data, what data they are requesting, how old the data is, what Internet domain or IP address they are requesting from, as well as the current date and time.

The two Enforce Privacy Tags operators are used to put the preferences specified in privacy tags into action. The out-operator version makes sure that data that should not leave an infospace does not, while the in-operator version does the same with incoming data. Together, a set of infospaces can provide peer enforcement of privacy tags, helping to ensure that data is managed properly (see Figure 3). Assuming that tuples are digitally signed, peers can also detect if privacy tags have been altered, thus detecting that an infospace is not handling personal information properly. However, this feature is not yet implemented in the current implementation of Confab.

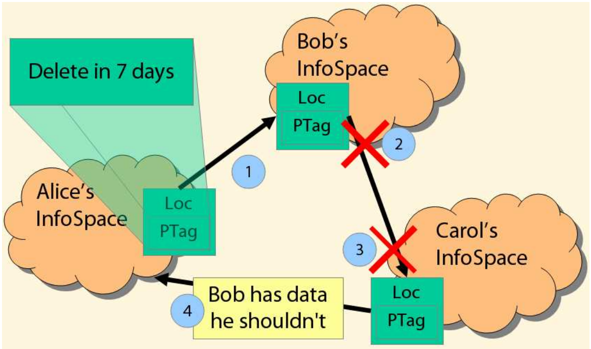


Figure 3. An example of peer enforcement. (1) Alice shares her location data with Bob. This data has been tagged to be deleted in seven days. Suppose seven days have passed, and that Bob passes the data on to Carol. If this is an accidental disclosure, then (2) his infospace prevents this from occurring. If this is intentional, then (3) Carol can detect that Bob has passed on data that he should not have, and (4) notifies Alice.

The Notify operators are used to send short messages to give end-users feedback about who is requesting information and when. Notify operators can currently be configured to send messages either through email or via instant messenger.

The Invisible mode operator can be used to block all outgoing tuples and return the value of "UNKNOWN" to all queries. The Invisible mode operator can also be configured to return some pre-specified value, allowing users to make "white lies". The Add Privacy Tag operator is used to add end-user or application defined privacy tags on outgoing tuples.

The Interactive operator can be used to give end-users control over disclosures. In the current implementation, when a request comes in and the Interactive operator is active, a simple GUI is displayed, giving the end-user several options, including disclosing the requested information just this once, ignoring it, or denying access permanently. An example of this user interface is shown in Figure 6.

The Garbage Collector operator is run periodically to delete data that have privacy tags that specify that they should be deleted. The Periodic Report operator sends an email to the owner of an infospace, providing a periodic summary of who has requested what (e.g., every day, week, or month). The Coalesce operator is used to delete tuples with repeated values. For example, suppose a user has a sensor that updates her infospace with her current location information every minute. If she has not moved for an hour, there will be sixty tuples with the exact same location value. Here, the Coalesce operator sorts all of the location tuples by time and deletes tuples with duplicate values, keeping only those needed to determine when she entered and exited a location.

Operators are loaded through a configuration file on startup, and are executed according to the order in which they were added. Each operator also has a filter that checks whether or not it should be run on a specific tuple. When an in- or out-method is called, a chain of the appropriate operators is assembled and then run on the set of incoming or outgoing tuples.

```
<Service    name="Tourguide"
     description="Tourguide for cities"
        keywords="Tourism, Location"
        provider="Bob Inc"
             url="http://bob.com/tourguide"
         version="1.0">

  <Option   name="1"
      dataformat="city"
        datatype="location"
          method="get"
           offer="Events, Museum lines"
            rate="15 minutes"
        timespan="current" />

  <Option   name="2"
      dataformat="zipcode"
        datatype="location"
          method="get"
           offer="Stores, Recommendations"
            rate="30 seconds"
        timespan="current" />

  <Option   name="3"
      dataformat="latlon"
        datatype="location"
          method="get"
           offer="Route Finder, Real-time map"
            rate="30 seconds"
        timespan="current" />

</Service>
```

Figure 4. Confab's service descriptions allow services to give end-users various choices when using a service. This example shows the service description for a mobile tour guide service. The first option (where name="1") provides information about events and the length of museum lines in the city. To do this, the service needs the end-user's current location at the city level every 15 minutes.

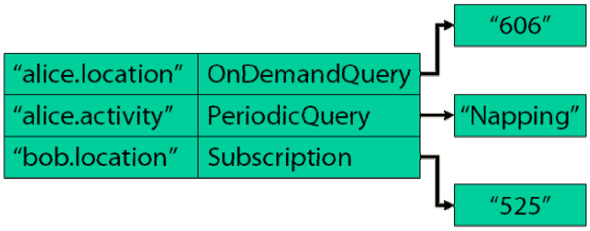


Figure 5. Clients can maintain a list of properties they are interested in through an Active Properties object, which will automatically issue queries and maintain last known values.

Note that peer enforcement and automatic deletion of old data can be trusted to execute on computers that an end-user has control over, but not necessarily on computers owned by others. Short of a trusted computing base, there is no way of forcing others to delete data. Privacy tags let end-users provide a hint saying what their privacy preferences are, and relies on social, legal, and market mechanisms that others will do the right thing. In cases where there is a strong level of trust, this will suffice and can help prevent accidental disclosures. In cases where there is not a great deal of trust, other mechanisms, such as passing on coarser-grained data or anonymity, should be used.

Service Descriptions

Applications can publish service descriptions that describe the application, as well as various options that end-users can choose from. For example, Scenario 2 described a mobile tour guide service that offered different kinds of information depending on the precision of information Alice was willing to share.

Confab provides support for applications to specify these different options, as shown in Figure 4. These service descriptions provide basic information about the service, for example the name of the service and a URL for more information. Service descriptions can also contain options that describe what features that option offers, what datatypes and dataformats are needed from the end-user, and how often the information will be queried.

When a client application first makes a request to an infospace, it sends its service description. If the infospace has seen this service description before, it simply uses the previously stored configuration associated with that description, which specifies whether to allow access and what option to use. If the infospace has not seen this service description before or the previous settings have expired, a default GUI is displayed which lets end-users choose whether to allow access, what option they want, and how long the settings should last. This approach gives service providers a way of giving end-users flexibility over what features they are interested in using as well as what privacy tradeoffs they are willing to make.

Active Properties

To simplify the task of querying for and maintaining context state in applications, Confab provides an *active properties* object for clients (see Figure 5). Queries can be placed in an active properties instance and be periodically executed to get up-to-date values. Last known values are also automatically stored, to provide a level of fault-tolerance should the requestor or requestee be temporarily disconnected.

Active properties supports three different kinds of properties: OnDemandQuery, which makes a request for new data whenever its value is checked; PeriodicQuery, which periodically checks for new data; and Subscription, which periodically receives new data from an infospace. After initial setup, clients can simply query the active properties using the property name (e.g., "alice.location") to retrieve the last-known value.

Summary

In summary, Confab's data model and programming model provide application developers with a framework and a suite of mechanisms for building privacy-sensitive applications. Operators are used within an end-user's infospace to help control the flow of personal information, and can be customized to fit specific end-user needs. Service descriptions are used by applications to describe what kinds of personal information are needed, as well as at what granularity and at what rate. Active properties are used by applications to automatically query and maintain contextual information about entities, simplifying programming for clients.

3.4 Extensions for Location Privacy

Since location-enhanced applications are a rapidly emerging area of ubiquitous computing, Confab currently comes with specific extensions for capturing and processing location information. In this section, we describe the Place Lab sensor source and the MiniGIS operator for processing location information.

Place Lab [63] uses the wide deployment of 802.11b WiFi access points for determining one's location in a privacy-sensitive manner. The key observation here is that many developed areas have wireless hotspot coverage so dense that cells overlap. By keeping a local cache of a Place Lab directory, which maps the unique MAC address of a wireless hotspot to a physical latitude and longitude, mobile computers and PDAs equipped with WiFi can determine their location to within a city block.

This approach can be done without any special equipment other than a WiFi card, and also works indoors and in urban canyons, places where GPS does not always work effectively. Furthermore, since wireless hotspots can be detected passively, computers can determine their location without divulging any information to any third parties or other entities.

Place Lab is currently implemented as a sensor source within Confab, adding a tuple representing the user's current latitude and longitude into the user's infospace every 60 seconds. Our current working database of WiFi access points for the San Francisco Bay Area (including the cities of San Francisco, Oakland, Berkeley, Palo Alto, and San Jose) has roughly 60000 nodes contained in about 4 megabytes of data, making it feasible to store on PDAs and laptops.

The MiniGIS operator transforms location information from one datatype to another locally on one's computer, for example from the latitude and longitude "37.7,-122.68" to the city name "San Francisco". This is useful for two reasons. The first is because latitude and longitude are difficult to comprehend and need to be put in a format semantically meaningful to people. The second is that MiniGIS does this transformation locally without disclosing any information to equivalent network services (such as Microsoft's MapPoint[1]).

MiniGIS currently has several built-in location datatypes, including latitude and longitude, place name ("Soda Hall"), city name, ZIP Code, region name ("California") and region code ("CA"), as well as country name ("United States") and country code ("USA"). MiniGIS can also be used to return the distance between two latitude and longitude pairs, as well as query for nearest locations, such as nearest places and cities.

MiniGIS is built from public data sources from the USGS[2] and GeoNET[3], and has roughly 30 megabytes of data. We have also been manually collecting data for place names using a GPS system, gathering the names of local cafes, landmarks, and other points of interest.

3.5 Implementation

Confab is implemented in Java 2 v1.5, and is currently comprised of 550 classes and approximately 55,000 physical lines of code (not including comments and boilerplate). Confab uses HTTP for network communication and is built on top of the Tomcat web server, making extensive use of Java servlets. XPath is used as the query language for matching and retrieving XML tuples, with Jaxen as the specific XPath engine.

The Place Lab sensor source is comprised of 10 classes and 1700 lines of code. MiniGIS is comprised of 15 classes and 3300 lines of code. Both Place Lab and MiniGIS make use of the MySQL open source database.

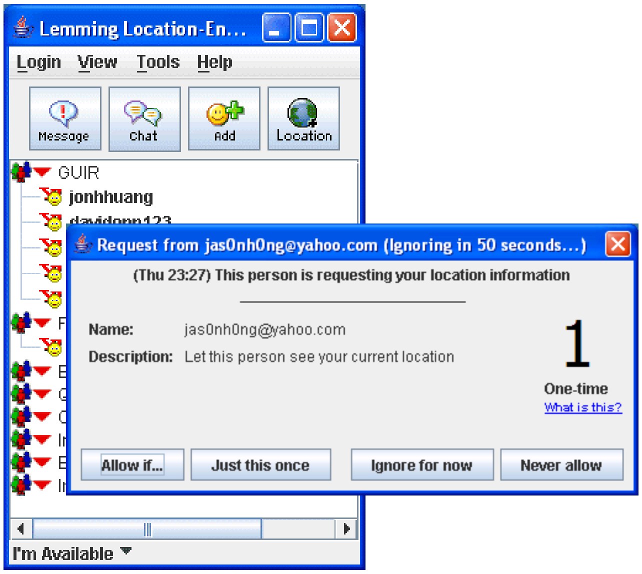
Confab also comes with a microphone source, which is used to estimate activity level, as well as several web-based simulators for faking location and activity data using a web browser.

4. EVALUATION

In this section, we describe the implementation of three applications we have built on top of Confab.

App #1 – Lemming Location-Enhanced Instant Messenger

Using Confab, we have built Lemming, a new location-enhanced instant messenger client that provides two features in addition to standard clients. The first new feature is the ability to request a user's current location (see Figure 6). When a location request is received, the end-user can choose "Never allow" to never allow the requestor to see her location, "Ignore for now" to ignore this current request (the default), "Just this once" to allow the request just this once, or "Allow if…" to always allow requests under certain conditions, such as from 9AM to 5PM or only between Monday and Friday.


Lemming Location-En...

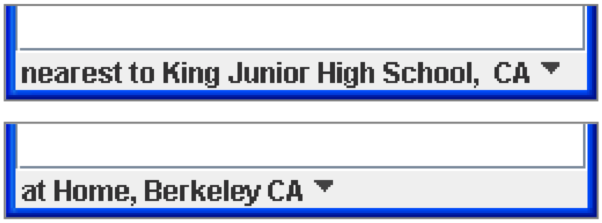
Request from jas0nh0ng@yahoo.com (Ignoring in 50 seconds…)

(Thu 23:27) This person is requesting your location information

Name: jas0nh0ng@yahoo.com

Description: Let this person see your current location

Figure 6. Lemming is a location-enhanced messenger that lets users query each other for their current location information. This screenshot shows the UI that lets a requestee choose whether or not to disclose their current location. The large "1" on the side represents that this is a one-time disclosure rather than a continuous disclosure of location information.


nearest to King Junior High School, CA

at Home, Berkeley CA

Figure 7. This location-enhanced messenger lets users set an away message describing their current location, which automatically updates as they move around.

[1] http://mappoint.msn.com

[2] http://geonames.usgs.gov/stategaz/index.html

[3] http://earth-info.nima.mil/gns/html/

From a software architecture perspective, when a location request is received, the end-user's instant messenger client issues a query to her infospace for her current location. Currently, Confab does not provide mechanisms for authentication, relying instead on the application itself to manage it. The infospace checks if there is a context tuple representing location information, and then checks the age of the tuple to see if it can be considered "current" (by default, this is set to twenty minutes).

At this point, the tuple flows through the out-operators defined in the infospace. The three operators of interest here are the Enforce Access Policies, Interactive, and MiniGIS operators. The Enforce Access Policies operator checks if there is an existing policy associated with the requestor and applies that policy if it exists. The Interactive operator also checks if there is an existing policy, and if there is not, brings up the user interface shown in Figure 6, letting end-users set a policy. Lastly, the MiniGIS operator runs, transforming the data from "latitude and longitude" into "place".

The second new feature is the ability to automatically display one's current location as an away message that automatically updates itself as one's location changes (see Figure 7). The Lemming instant messenger client sets up a query to get the nearest "place" every 60 seconds, and then displays this place as the away message. Lemming currently defines three place descriptions based on the user's distance to that place: "at", if the distance is less than 10 meters; "near", if the distance is less than 100 meters; and "nearest to" if the distance is greater than 100 meters.

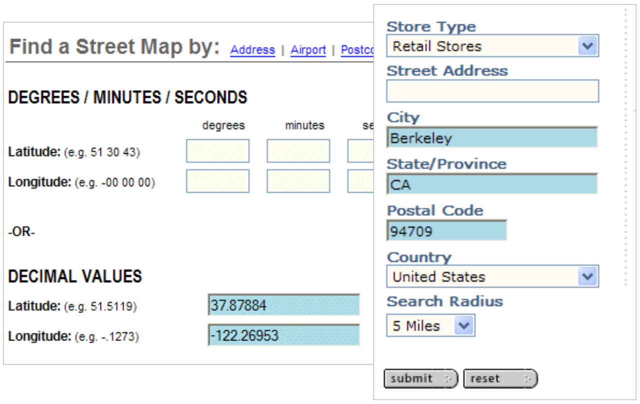
Lemming uses the Hamsam library for cross-platform instant messaging[4]. Lemming is roughly 2500 lines of code across 23 classes. It took about 5 weeks to build, with the majority of the effort and code devoted to the GUI. Here, Confab provides support for acquiring location information, storing location information and privacy preferences, making location queries, automatically updating location information for the away message, and MiniGIS for processing location information.

App #2 – Location-Enhanced Web Proxy

We have also built a location-enhanced web proxy that can automatically fill in fields on web sites (see Figure 8). When the proxy is started, it loads up a configuration file that describes which URLs to look for, which HTML input fields to modify, and what values to insert in those fields. Some possible values include one's current city, state, ZIP Code, and latitude and longitude.

Users can run this proxy locally on their computer and set their web browser to use this proxy. Whenever the proxy detects one of the pre-defined URLs, it modifies the HTML, inserting the current location information and highlighting the modified fields in blue. To help protect the privacy of the user, the proxy is restricted to accept connections only from localhost.

The location-enhanced web proxy is roughly 800 lines of code, added to an existing base of 800 lines of code from an open-source web proxy. It took about one week to build. Here, Confab provides support for making location queries for one's current location, automatically updating one's location, as well as MiniGIS for processing location information.



Find a Street Map by: Address | Airport | Postco

DEGREES / MINUTES / SECONDS

	degrees	minutes	se
Latitude: (e.g. 51 30 43)			
Longitude: (e.g. -00 00 00)			

-OR-

DECIMAL VALUES

Latitude: (e.g. 51.5119) 37.87884

Longitude: (e.g. -.1273) -122.26953

Store Type: Retail Stores

Street Address:

City: Berkeley

State/Province: CA

Postal Code: 94709

Country: United States

Search Radius: 5 Miles

Figure 8. The location-enhanced web proxy can automatically fill in fields requesting location information on web pages. The page on the left is from MapQuest (http://mapquest.com), with latitude and longitude automatically filled in. The page on the right is a store finder from StarBucks (http://starbucks.com), with city, state/province, and postal code automatically filled in.

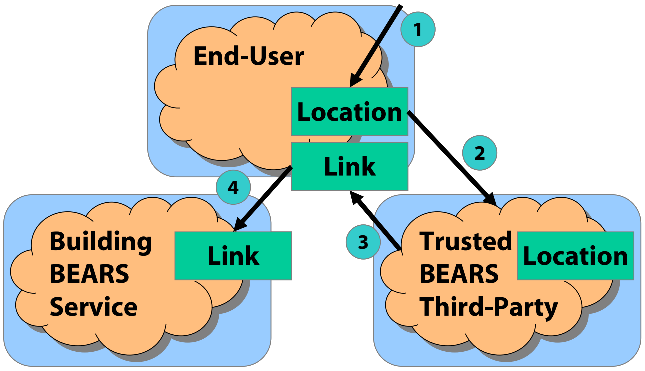


Figure 9. An example setup of the BEARS emergency response service. First, an end-user obtains their location (1) and shares it with a trusted third-party (2). The end-user gets a link (3) that can be sent to others, in this case to a building (4). If there is an emergency, responders can traverse all known links, getting up-to-date information about who is in the building (with the trusted third-party notifying data sharers what has happened).

App #3 – BEARS Emergency Response Service

One emerging application for location-enhanced phones is Enhanced 911. E911 lets users share their location with dispatchers when making emergency calls on mobile phones. One's location is only transmitted to dispatchers when the call is actually made. While there are many advantages to E911, one downside is that it is a discrete push system. There are no easy ways of getting a person's current or last-known location in known emergencies, for example, an earthquake, a building fire, or a kidnapping.

BEARS is a system we are developing to handle these cases. There are two tensions to balance here. On the one hand, we want location information to be highly available in the case of emergencies. On the other, emergencies are rather rare, and so we also want some guarantees that location information will be used exclusively for emergencies and for no other purposes.

[4] http://hamsam.sourceforge.net/

BEARS works by having a trusted third-party store one's location information in case of emergencies. This third party can be a friend or even a paid service whose business model is predicated on providing location information only in the event of emergencies. Such services already exist with respect to one's medical information, the best known of which is MedicAlert [3]. These services would have a significant market incentive to use location information only for stated purposes and possibly a legal obligation as well.

Figure 9 shows an example of how BEARS can be used in buildings to keep track of who is in the building and where they are for emergency response purposes. First, an end-user obtains his location. He periodically sends his location to the trusted third party, which gives him one or more named links back to this data. The end-user can then share this link with others, such as a building. In case of emergencies, the link can be traversed, with last-known location information being retrieved from the third party, with the third party also notifying end-users that their information has been discloesd. This approach allows emergency responders to get critical location information, provides a level of redundancy should the user's device or location systems fail or if the end-user is incapacitated, and provides a basic level of privacy.

The BEARS client is roughly 200 lines of code and took about 2 days to create. The reason for its small size is that there is no GUI. Here, Confab provides support for making continuous location queries, as well as making updates to both the trusted third-party and to the building server.

We have also used Confab to build prototypes of applications that have minimal privacy concerns. One that is currently in progress is emergency response support to help firefighters on scene. Our prototype uses sensors and PDAs to automatically gather and disseminate information about the fire and about that firefighter to other nearby firefighters [40]. Another is a distributed querying system for supporting database operations, such as join or project, for streaming data and across multiple infospaces [36].

5. RELATED WORK

There has been a great deal of work at providing programming support for various aspects of ubiquitous context-aware computing. This includes the PARCTab system [61], Cooltown [44], the Context Toolkit [20], Contextors [17], Limbo [19], Sentient Computing [6], Stick-E notes [56], MUSE [15], SpeakEasy [23], Solar [16], XWeb [53], GAIA [60], one.world [30], and iRoom [42]. Confab builds on this previous work, with the key difference being that Confab's architecture and mechanisms are focused on helping application developers and end-users manage personal privacy.

Confab is closest in terms of data model and programming model to the PARCTab system [61] and iRoom [42]. In many ways, Confab's data model can be thought of as a logical evolution of the PARCTab's Dynamic Environments. Dynamic Environments are centralized data stores associated with relatively large places, such as buildings. Each Dynamic Environment contains personal information about people, places, and things within its purview. As people move from place to place, they also switch which Dynamic Environment they are using. The key differences Confab makes are decentralization of data so that personal information is stored and processed on the end-user's computer as much as possible, a greater range of mechanisms for privacy in both the data model and in the programming model, and compartmentalized extensibility thru operators.

The iRoom is a suite of software to support interactive workspaces at the room level. Central to this is the EventHeap, a shared tuplespace for the room in which input devices can place events and output devices can receive events. This level of indirection encourages looser coupling between application components and fosters greater overall robustness. Confab uses a similar approach with its infospaces, separating sources of data (such as sensors) from the services and applications that use them, with little or no knowledge of each other. Like the EventHeap, Confab also has a thin API with few methods. The main difference between the EventHeap and Confab is that Confab is specialized for building privacy-sensitive systems. Confab also looks at supporting multiple infospaces to represent people, places, and things, rather than just one tuplespace to represent all events and information within a place. Again, Confab takes a decentralized approach, placing information about end-users on their computers as much as possible.

There has also been some previous work on using digital rights management in managing personal information. Langheinrich [47] described pawS, a privacy awareness system for ubicomp that lets deployed systems announce P3P policies of what data is being collected, and offers database support for enforcing those policies. Similarly, IBM has also introduced an Enterprise Privacy Authorization Language [39] that lets developers describe privacy policies and attach those privacy policies to the data as it flows through a company. The privacy tags in Confab are similar in spirit to these ideas, while also introducing further digital rights management ideas, such as using location as a parameter, enforcing a maximum number of past sightings, and peer enforcement.

Confab also builds on the work by Spreitzer and Theimer [65], who describe an architecture for providing location information. In their architecture, each user owns a User Agent that collects and controls all personal information pertaining to its user, and any request for such information must be routed through the User Agent which enforces predetermined access policies. Confab takes this same basic approach and extends it with a wider range of privacy mechanisms, including notifications, tags, logging, and interactive requests, to support the development of pessimistic, optimistic, and mixed-initiative type applications.

There has been a great deal of work in providing levels of anonymity in networked systems. One system of note here is Gruteser and Grunwald's work on spatial and temporal cloaking [34], in which a trusted proxy is used to adjust the resolution of location reported to services based on the density of users in a region. Since many users report their location through the proxy, user density is known. Thus, the proxy can provide k-anonymity, that is hiding one's precise location by returning an area that has k-1 other people. Sweeney [66] has proposed a general approach for doing k-anonymity for static database tables, aggregating data together into buckets to reduce identifiability. Another approach is to use mixes to make it harder to do traffic analysis (e.g. [10]). Confab currently does not have any built-in support for managing anonymity or for defeating traffic analysis, but could support these approaches in its architecture. Confab also provides support for application in which anonymity is not useful, for example, situations with family, friends, co-workers, and paid services.

In summary, while there have been many toolkits and infrastructures providing programming support and abstractions for sensors, and while there have been many individual techniques for managing privacy, Confab is the first to provide an extendable design that provides software architecture support for building privacy-sensitive ubicomp applications that are optimistic, pessimistic, and mixed-initiative. Confab provides reusable mechanisms for both application developers and for end-users in managing personal information, as well as mechanisms and abstractions for developers designing privacy-sensitive ubicomp systems.

6. FUTURE WORK

We plan on building more ubicomp applications on top of Confab, and are currently in the process of evaluating the applications described above with real users to assess how well people can understand the basic model of what the system knows about them and where their information is flowing, the privacy implications in sharing personal information, and the overall ease of interaction.

7. CONCLUSIONS

We presented an extensive analysis of end-user needs and application developer needs for privacy-sensitive systems. The end-user needs were gathered through scenario-based interviews we did on location-enhanced applications, and by an analysis of surveys, research papers, message boards, proposed and existing privacy protection laws, and design guidelines for privacy-sensitive systems. The application developer needs were gathered through an analysis of research and commercial ubicomp applications.

These needs led to the high-level requirements of (1) a decentralized architecture, (2) a range of control and feedback mechanisms for building pessimistic, optimistic, and mixed-initiative applications, (3) plausible deniability built in, and (4) exceptions for emergencies.

To address these needs, we developed Confab, a toolkit for building privacy-sensitive ubicomp applications for a spectrum of trust levels and privacy needs. From a software architecture perspective, Confab provides a framework and an extendable suite of mechanisms that application developers and end-users can use for managing privacy. This framework was designed such that personal information is captured, stored, and processed on the end-user's computer as much as possible. This provides end-users with a greater amount of choice and control than previous systems over what information they wish to share with others.

We also illustrated how Confab was used to support the implementation of three privacy-sensitive ubicomp applications, including a location-enhanced instant messenger, a location-enhanced web proxy, and an emergency response application.

8. ACKNOWLEDGMENTS

Thanks to Leendert Van Doorn (our shepherd) and our reviewers for valuable comments and suggestions. Chris Beckmann, Jeff Heer, and Alan Newberger designed and implemented the liquid distributed querying system on top of Confab. Jennifer Ng worked on the end-user interviews and on transcribing audio notes, and worked with Eric Chung and Madhu Prabaker in collecting place names and WiFi data around the San Francisco Bay Area. Special thanks to Gregory Abowd, Gaetano Boriello, John Canny, Anind Dey, Xiaodong Jiang, Scott Lederer, Bill Schilit, Doug Tygar, and Terry Winograd, as well as the participants in many privacy workshops for feedback on refining the ideas in this paper. This work has been supported in part by NSF (IIS-0205644), DARPA (N66001-99-2-8913), an Intel fellowship, a Siebel Scholar fellowship, and PARC.

9. REFERENCES

[1] AllNurses.com. http://allnurses.com/

[2] Directive 95/46/EC. http://europa.eu.int/ISPO/legal/en/dataprot/directiv/directiv.html

[3] MedicAlert. http://www.medicalert.org

[4] Abowd, G.D., C.G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, Cyberguide: A Mobile Context-Aware Tour Guide. *Baltzer/ACM Wireless Networks* 1997. **3**(5): p. 421-433.

[5] Adams, A. Multimedia Information Changes the Whole Privacy Ball Game. In Proceedings of *Computers, Freedom, and Privacy*. Toronto, Canada: ACM Press. pp. 25-32 2000.

[6] Addlesee, M., R. Curwen, S.H. Newman, P. Steggles, A. Ward, and A. Hopper, Implementing a Sentient Computing System. *IEEE Computer* 2001. **34**(8): p. 50-56.

[7] AT&T, AT&T Wireless mMode - Find Friends. http://www.attwireless.com/mmode/features/findit/FindFriends/

[8] Barkhuus, L. and A.K. Dey. Location-based services for mobile telephony: a study of users' privacy concerns. In Proceedings of *INTERACT 2003, 9th IFIP TC13 International Conference on Human-Computer Interaction*. pp. To appear 2003.

[9] Bellotti, V. and A. Sellen. Design for Privacy in Ubiquitous Computing Environments. In Proceedings of *The Third European Conference on Computer Supported Cooperative Work (ECSCW'93)*. Milan, Italy: Kluwer Academic Publishers 1993.

[10] Beresford, A. and F. Stajano, Location Privacy in Pervasive Computing, *IEEE Pervasive Computing*, vol. 2(1): pp. 46-55, 2003.

[11] Brin, D., *The Transparent Society*. Reading, MA: Perseus Books, 1998.

[12] Brown, P.J. and G.J.F. Jones, Context-aware Retrieval: Exploring a New Environment for Information Retrieval and Information Filtering. *Personal and Ubiquitous Computing* 2001. **5**(4): p. 253-263.

[13] Burrell, J., G.K. Gay, K. Kubo, and N. Farina. Context-Aware Computing: A Test Case. In Proceedings of *Ubicomp 2002*. Göteborg, Sweden. pp. 1-15 2002.

[14] Cadiz, J. and A. Gupta, *Privacy Interfaces for Collaboration*. Technical Report MSR-TR-2001-82, Microsoft Research, Redmond, WA 2001.

[15] Castro, P. and R. Muntz, Managing Context for Smart Spaces. *IEEE Personal Communications* 2000. **5**(5).

[16] Chen, G. and D. Kotz. Context Aggregation and Dissemination in Ubiquitous Computing Systems. In

Proceedings of *Fourth IEEE Workshop on Mobile Computing Systems and Applications*. pp. 105-114 2002.

[17] Crowley, J.L., J. Coutaz, G. Rey, and P. Reignier. Perceptual Components for Context Aware Computing. In Proceedings of *Ubicomp 2002*. Göteborg, Sweden. pp. 117-134 2002.

[18] Cuellar, J., J. John B. Morris, D. Mulligan, J. Peterson, and J. Polk, Geopriv requirements (Internet Draft). 2003, IETF. http://www.ietf.org/internet-drafts/draft-ietf-geopriv-reqs-04.txt

[19] Davies, N., S.P. Wade, A. Friday, and G.S. Blair. Limbo: A tuple space based platform for adaptive mobile applications. In Proceedings of *The International Conference on Open Distributed processing / Distributed Platforms (ICODP/ICDP '97)*. pp. 291-302 1997.

[20] Dey, A.K., D. Salber, and G.D. Abowd, A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction (HCI) Journal* 2001. **16**(2-3): p. 97-166.

[21] Doheny-Farina, S., The Last Link: Default = Offline, Or Why Ubicomp Scares Me, *Computer-mediated Communication*, vol. 1(6): pp. 18-20, 1994.

[22] Edwards, J., Location Privacy Protection Act of 2001. http://www.techlawjournal.com/cong107/privacy/location/s1164is.asp

[23] Edwards, W.K., M.W. Newman, J.Z. Sedivy, T.F. Smith, and S. Izadi. Challenge: Recombinant Computing and the Speakeasy Approach. In Proceedings of *Eighth ACM International Conference on Mobile Computing and Networking (MobiCom 2002)*. pp. 279-286 2002.

[24] Espinoza, F., P. Persson, A. Sandin, H. Nyström, E. Cacciatore, and M. Bylund. GeoNotes: Social and Navigational Aspects of Location-Based Information Systems. In Proceedings of *Ubicomp 2001*. Atlanta, GA. pp. 2-17 2001.

[25] Falk, J., P. Ljungstrand, S. Björk, and R. Hansson. Pirates: Proximity-Triggered Interaction in a Multi-Player Game. In Proceedings of *Human Factors in Computing Systems: CHI 2001 (Extended Abstracts)*. pp. 119-120 2001.

[26] Federal Communications Commission, Enhanced 911. http://www.fcc.gov/911/enhanced/

[27] Frelinghuysen, R., Wireless Privacy Protection Act of 2003. http://www.theorator.com/bills108/hr71.html

[28] Garfinkel, S., *Database Nation: The Death of Privacy in the 21st Century*: O'Reilly & Associates, 2001.

[29] Geocaching. http://www.geocaching.com/

[30] Grimm, R., J. Davis, E. Lemar, A. Macbeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, and D. Wetherall, *Programming for pervasive computing environments*. Technical Report UW-CSE-01-06-01, University of Washington Department of Computer Science and Engineering, Seattle, WA 2001.

[31] Griswold, W.G., P. Shanahan, S.W. Brown, and R. Boyer, *ActiveCampus - Experiments in Community-Oriented Ubiquitous Computing*. Technical Report CS2003-0765, Computer Science and Engineering, UC San Diego 2003.

[32] Grudin, J., Desituating Action: Digital Representation of Context. *Human-Computer Interaction (HCI) Journal* 2001. **16**(2-4).

[33] Grudin, J. and E. Horvitz, Presenting choices in context: approaches to information sharing. 2003: Workshop on Ubicomp communities: Privacy as Boundary Negotiation. http://guir.berkeley.edu/pubs/ubicomp2003/privacyworkshop/papers.htm

[34] Gruteser, M. and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In Proceedings of *The First International Conference on Mobile Systems, Applications, and Services (MobiSys 2002)* 2002.

[35] Harper, R.H.R., *Why Do People Wear Active Badges?* Technical Report EPC-1993-120, Rank Xerox, Cambridge 1993.

[36] Heer, J., A. Newberger, C. Beckmann, and J.I. Hong. liquid: Context-Aware Distributed Queries. In Proceedings of *Fifth International Conference on Ubiquitous Computing: Ubicomp 2003*. Seattle, WA: Springer-Verlag. pp. 140-148 2003.

[37] Hindus, D., S.D. Mainwaring, N. Leduc, A.E. Hagström, and O. Bayley, Casablanca: Designing Social Communication Devices for the Home. *CHI Letters (Human Factors in Computing Systems: CHI 2001)*, 2001. **3**(1): p. 325-332.

[38] Hong, J.I., G. Boriello, J.A. Landay, D.W. McDonald, B.N. Schilit, and J.D. Tygar. Privacy and Security in the Location-enhanced World Wide Web. In Proceedings of *Fifth International Conference on Ubiquitous Computing: Ubicomp 2003 (Workshop on Ubicomp Communities: Privacy as Boundary Negotiation)*. Seattle, WA 2003.

[39] IBM Corporation, Enterprise Privacy Authorization Language (EPAL 1.1). http://www.zurich.ibm.com/security/enterprise-privacy/epal/Specification/

[40] Jiang, X., N.Y. Chen, J.I. Hong, K. Wang, L.A. Takayama, and J.A. Landay. Siren: Context-aware Computing for Firefighting. In Proceedings of *The Second International Conference on Pervasive Computing (Pervasive 2004)*. Vienna, Austria. pp. To Appear 2004.

[41] Jiang, X., J.I. Hong, and J.A. Landay. Approximate Information Flows: Socially-based Modeling of Privacy in Ubiquitous Computing. In Proceedings of *Ubicomp 2002*. Göteborg, Sweden. pp. 176-193 2002.

[42] Johanson, B., A. Fox, and T. Winograd, The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. *IEEE Pervasive Computing* 2002. **1**(2): p. 67-74.

[43] Kaasinen, E., User Needs for Location-aware Mobile Services. *Personal and Ubiquitous Computing* 2003. **7**(1): p. 70-79.

[44] Kindberg, T. and J. Barton, A Web-based Nomadic Computing System. *Computer Networks* 2001. **35**: p. 443-456.

[45] Korba, L. and S. Kenny. Towards Meeting the Privacy Challenge: Adapting DRM. In Proceedings of *2002 ACM Workshop on Digital Rights Management*. Washington DC, USA 2002.

[46] Lamming, M. and M. Flynn. Forget-me-not: Intimate computing in support of human memory. In Proceedings of *FRIEND 21: International Symposium on Next Generation Human Interfaces*. Meguro Gajoen, Japan. pp. 125-128 1994.

[47] Langheinrich, M. A Privacy Awareness System for Ubiquitous Computing Environments. In Proceedings of *Ubicomp 2002*. Goteberg, Sweden. pp. 237-245 2002.

[48] Langheinrich, M. Privacy by Design - Principles of Privacy-Aware Ubiquitous Systems. In Proceedings of *Ubicomp 2001*. Atlanta, GA. pp. 273-291 2001.

[49] Lederer, S., J. Mankoff, and A.K. Dey. Who Wants to Know What When? Privacy Preference Determinants in Ubiquitous Computing. In Proceedings of *Extended Abstracts of CHI 2003, ACM Conference on Human Factors in Computing Systems*. Fort Lauderdale, FL. pp. 724-725 2003.

[50] Lessig, L. The Architecture of Privacy. In Proceedings of *Taiwan NET'98*. Taipei, Taiwan 1998.

[51] Mayor, M., New Wireless Device Could Rescue Firefighters. 2001. http://www.wirelessnewsfactor.com/perl/story/9134.html

[52] Nagel, K., C.D. Kidd, T. O'Connell, A. Dey, and G.D. Abowd. The Family Intercom: Developing a Context-Aware Audio Communication System. In Proceedings of *Ubicomp 2001*. Atlanta, GA. pp. 176-183 2001.

[53] Olsen, D.R., S. Jefferies, T. Nielsen, W. Moyes, and P. Frederickson, Cross-modal Interaction using XWeb. *CHI Letters, The 13th Annual ACM Symposium on User Interface Software and Technology: UIST 2000* 2000. **2**(2): p. 191-200.

[54] OnStar. http://www.onstar.com/

[55] Palen, L. and P. Dourish, Unpacking "Privacy" for a Networked World. *CHI Letters (Human Factors in Computing Systems: CHI 2003)*, 2003. **5**(1): p. 129-136.

[56] Pascoe, J. The Stick-e Note Architecture: Extending the Interface Beyond the User. In Proceedings of *International Conference on Intelligent User Interfaces*. pp. 261-264 1997.

[57] Povey, D. Optimistic Security: A New Access Control Paradigm. In Proceedings of *1999 New Security Paradigms Workshop* 1999.

[58] Priyantha, N.B., A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In Proceedings of *MobiCom 2000: The Sixth Annual International Conference on Mobile Computing and Networking*. Boston, Massachusetts: ACM Press. pp. 32-43 2000.

[59] Rhodes, B. and T. Starner. The Remembrance Agent: A Continuously Running Automated Information Retrieval System. In Proceedings of *The First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology (PAAM '96)*. London, UK. pp. 487-495 1996.

[60] Román, M., C.K. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt, Gaia: A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing* 2002. **1**(4): p. 74-83.

[61] Schilit, B.N., *A Context-Aware System Architecture for Mobile Distributed Computing*, Unpublished PhD, Columbia University, 1995. http://seattleweb.intel-research.net/people/schilit/schilit-thesis.pdf

[62] Schilit, B.N., N.I. Adams, and R. Want. Context-Aware Computing Applications. In Proceedings of *Workshop on Mobile Computing Systems and Applications*. Santa Cruz, CA: IEEE Computer Society, December 1994 1994.

[63] Schilit, B.N., G. Borriello, W.G. Griswold, D. McDonald, A. Lamarca, J. Hong, E. Lazowska, A. Balachandran, and V. Iverson. Challenge: Ubiquitous Location-Aware Computing. In Proceedings of *The First ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH '03)*. San Diego, CA: ACM Press. pp. To Appear 2003.

[64] Sloane, L., Orwellian Dream Come True: A Badge That Pinpoints You, *New York Times* pp. 14, 1992.

[65] Spreitzer, M. and M. Theimer. Providing location information in a ubiquitous computing environment. In Proceedings of *Fourteenth ACM Symposium on Operating System Principles*. Asheville, NC: ACM Press, December 1993.

[66] Sweeney, L., k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* 2002. **10**(5): p. 557-570.

[67] Talbott, S., The Trouble with Ubiquitous Technology Pushers, or: Why We'd Be Better Off without the MIT Media Lab. 2000. http://www.oreilly.com/people/staff/stevet/netfuture/2000/Jan0600_100.html

[68] Want, R., A. Hopper, V. Falcão, and J. Gibbons, The Active Badge Location System. *ACM Transactions on Information Systems* 1992. **10**(1): p. 91-102.

[69] Weiser, M., R. Gold, and J.S. Brown, The Origins of Ubiquitous Computing Research at PARC in the Late 1980s. *IBM Systems Journal* 1999. **38**(4): p. 693-696.

[70] Westin, A.F., *Privacy and Freedom*. New York NY: Atheneum, 1967.

[71] Whalen, J., You're Not Paranoid: They Really Are Watching You, *Wired Magazine*, vol. 3(3): pp. 95-85, 1995.