A Privacy Awareness System for Ubiquitous Computing Environments

Marc Langheinrich

Institute of Information Systems, ETH Zurich
8092 Zurich, Switzerland
www.inf.ethz.ch/~langhein

Abstract. Protecting personal privacy is going to be a prime concern for the deployment of ubiquitous computing systems in the real world. With daunting Orwellian visions looming, it is easy to conclude that tamper-proof technical protection mechanisms such as strong anonymization and encryption are the only solutions to such privacy threats. However, we argue that such perfect protection for personal information will hardly be achievable, and propose instead to build systems that help others respect our personal privacy, enable us to be aware of our own privacy, and to rely on social and legal norms to protect us from the few wrongdoers. We introduce a privacy awareness system targeted at ubiquitous computing environments that allows data collectors to both announce and implement data usage policies, as well as providing data subjects with technical means to keep track of their personal information as it is stored, used, and possibly removed from the system. Even though such a system cannot *guarantee* our privacy, we believe that it can create a sense of *accountability* in a world of invisible services that we will be comfortable living in and interacting with.

1 Motivation

It is undisputed that a future world full of smart and cooperating artifacts will pose great risks to our personal privacy: In an environment containing countless, invisible sensors that constantly monitor their surroundings and communicate their findings to each of their peers, both real-world and virtual transactions are certain to find their way into sheer limitless data storage systems, to be saved forever and recalled at a moment's notice. Much under discussion, however, is how to deal with this frightening vision. Should we give up on privacy as we know it today, and – at least potentially – make everybody see everything anytime [2]? Or should we instead try even harder to find the technological means that guarantee that our information stays private by employing state-of-the-art encryption and anonymization technology?

Even though clever anonymization technology [10] can make tracing our digital identity almost impossible, and encryption schemes exist that seem to be sufficiently hard to break in the foreseeable future, personal privacy entails more than just secret communication and masked identity. Unless we want to abandon our current social interactions completely and deal only behind digital pseudonyms in virtual reality with each other, we must realize that our real-world presence cannot be completely hidden, nor perfectly anonymized. Neither can postal addresses or names stored in a database

be protected from copying (like digital music) using digital watermarks, not unless we want to give up our (human-readable) first names, last name, and street names for large (machine-readable) binary representations that provide us with sufficient amount of information to embed such security features.

The privacy awareness system (*pawS*) presented here aims to strike a reasonable balance between those two approaches. It follows a fundamental principle used in today's democratic societies: to give people the ability to respect other people's safety, property, or privacy, and to rely on corresponding social norms, legal deterrence, and law enforcement to create a reasonable expectation that people will follow such rules. Examples for such inherently unsafe, yet trusted mechanisms are road traffic (where we don't directly regulate how someone drives but instead punish those who fail to follow the rules) or the local newspaper stand (where it might be possible to just pick up a paper without paying but where we rely on law enforcement to catch the thief eventually). Road signs and a monetary system are in these cases the mechanisms that allow people to respect other people's safety and property, yet do not guarantee that they are being used properly. Similarly, *pawS* provides *collection* and *processing* tools that allow data collectors and processors to communicate their collection and procession details to us, and help them keep their promises. While in individual cases more protection might be required (e.g., for sensitive data such as health records), most situations of our daily life should be adequately "protected" through such tools and corresponding enforcement and recourse mechanisms that allow holding people *accountable* to their public statements and actions.

Based on these assumptions, section 2 will present a short overview of our system, followed by the list of design requirements underlying its architecture. Section 3 then gives brief details on the current prototype *pawS* implementation, followed by a summary and pointers to future work in section 4.

2 General principle and requirements

Figure 1 on the next page shows an example of *pawS* in operation: Upon entering a ubicomp environment with a number of available services (here: a print service and a location tracking service using a video camera), a *privacy beacon* (1) announces the data collections of each service and their policies using a wireless communications channel such as Bluetooth or IrDA. In order to save energy, the mobile *privacy assistant* (2) the user is carrying delegates this information to the user's personal *privacy proxy* residing somewhere on the Internet (3), which contacts the corresponding service proxies at their advertised addresses (4) and inquires their *privacy policies*. After comparing those privacy policies to the user's privacy preferences, the user proxy decides to decline usage of the tracking service, which results in disabling the location tracking service of the video camera (5).

In designing the general architecture of such a privacy awareness system, we followed six principles set out earlier for preserving privacy in ubiquitous computing [8]: notice, choice and consent, proximity and locality, anonymity and pseudonymity, security, and access and recourse. As pointed out in the introduction, anonymity, pseudonymity, and security (i.e., secure communication and access) are useful tools

when being a supportive part of the infrastructure, but should not be taken as isolated solutions. Consequently, our system employs anonymous and secure connections, as well as reasonable access controls, whenever possible to prevent unwanted data spills and trivial data sniffing. Our main focus, however, lies on implementing the other four principles for use in a ubiquitous computing (ubicomp) environment:

- **Notice:** Given a ubicomp environment where it is often difficult for data subjects to realize that data collection is actually taking place, we will not only need mechanisms to declare collection practices (i.e., *privacy policies*), but also efficient ways to communicate these to the user (i.e., *policy announcement*).
- **Choice and consent:** In order to give users a true choice, we need to provide a selection mechanism (i.e., *privacy agreements*) so that users can indicate which services they prefer.
- **Proximity and locality:** The system should support mechanisms to encode and use *locality information* for collected data that can enforce access restrictions based on the location of the person wanting to use the data.
- **Access and recourse:** Our system needs to provide a way for users to access their personal information in a simple way through standardized interfaces (i.e., *data access*). Users should be informed about the usage of their data once it is stored, similar to call-lists that are often part of monthly phone bills (i.e., *usage logs*).

The following sections describe the four core concepts of our system, which provide us with the necessary functionality to implement the high-level requirements listed above: Machine-readable privacy policies to provide *choice and consent*, policy announcement mechanisms to give *notice*, privacy proxies for supporting *access*, and privacy-aware databases for *recourse*. While *proximity and locality* are not yet ad-

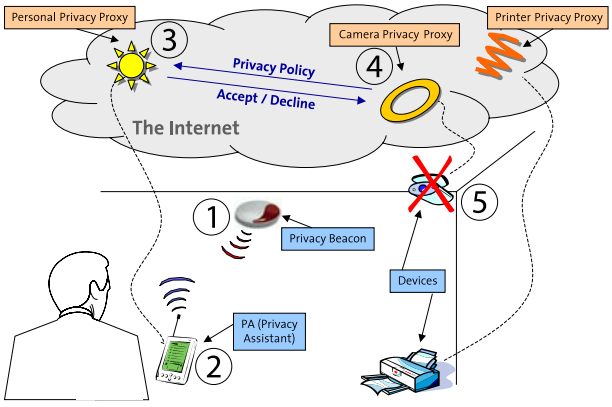

Fig. 1. Overview of the Privacy Management System: Upon entering a ubicomp environment with a number of data collections taking place (3,4), optional services can be configured to suit the user's privacy preferences (5). See section 2 for operation details. Mandatory data collections (e.g., security cameras) can at least be detected (1) and collection details be recorded (2), allowing users or consumer interest groups to hold data collectors accountable for their statements.

dressed in the current prototype, extension mechanisms allow for their implementation once suitable representation techniques have been developed.

2.1 Machine-readable privacy policies

Privacy policies are an established principle in legal domains to codify data collection and usage practices. Within the "Platform for Privacy Preferences Project (P3P)," the World Wide Web Consortium (W3C) recently finalized work that allows the encoding of such privacy policies into machine-readable XML, allowing automated processes to read such policies and take actions on them [6]. Figure 2 shows an abbreviated example of such a P3P privacy policy. It contains XML elements to describe for example who is collecting information (line 2, abbreviated), what data is being collected (lines 15-18), for whom (line 13), and why (line 12). For a more detailed explanation of the XML syntax see [6]. Using a similarly machine-readable preference language such as APPEL [5], users can express personal preferences over all aspects of such policies and have automated processes judge the acceptability of any such policy, or prompt for a decision instead. Since it might be cumbersome to manually create such preferences from scratch, a trusted third party (e.g., a consumer interest group) could provide pre-configured preference specifications that would then be downloaded and individually adjusted by each user.

Together with some domain-specific extension (e.g., location), these mechanisms allow data collectors in a ubicomp environment to specify data collection, storage and distribution parameters that can be automatically processed by user clients (*choice and consent*). It is important to note that typical environments will involve a reasonably

```
01: <POLICY name="FollowMe" discuri="http://www.example.org/services/follow-me/">
02:   <ENTITY> ...  </ENTITY>
03:   <DISPUTES-GROUP> ... </DISPUTES-GROUP>
04:   <ACCESS><all/>
05:     <EXTENSION optional="yes"> <ACCESS-METHODS>
06:       <UPDATE rpc_uri="http://www.example.org/soap/" service_urn="access">
07:         <DATA ref="#user.login.password"/> </UPDATE> </ACCESS-METHODS>
08:     </EXTENSION>
09:   </ACCESS>
10:   <STATEMENT>
11:     <CONSEQUENCE>Your telephone calls will be routed to you.</CONSEQUENCE>
12:     <PURPOSE><current/></PURPOSE>
13:     <RECIPIENT><ours/></RECIPIENT>
14:     <RETENTION><stated-purpose/></RETENTION>
15:     <DATA-GROUP> <DATA ref="#user.login.id"/>
16:                  <DATA ref="#user.login.password"/>
17:                  <DATA ref="#user.location.current.symbolic.room">
18:     </DATA-GROUP>
19:   </STATEMENT>
20: </POLICY>
```

Fig. 2. Example of a P3P policy for a follow-me telephone service (abbreviated, including access extensions): Apart from the user's id and password that has to be submitted when trying to use the service (lines 15-16), the service also (implicitly) collects the user's current location (e.g., room number) through a tracking system (line 17). An extension to the regular P3P syntax additionally describes the proxy access to the collected data (lines 5-8). See [6] for details.

small number of policies, even though a large number of sensors and data exchanges might be present, since policies are typically on a per realm or task basis. This means that the setup of a ubicomp environment with P3P policies is quite feasible.

2.2 Policy announcement mechanisms

While P3P is a Web technology and thus uses HTTP-headers as well as well-known URI-locations on each Web server to help user clients locate such policies, we need an alternative mechanism in a ubicomp environment. We can differentiate between two types of data collection that will need different ways of communicating such privacy policies to the data subject (*notice*):

- **Implicit announcement:** In many cases, the user client is actively locating and using a service offered by the environment. In this case, we embed links to the P3P policy (or even the policy itself) into the service discovery protocol, such as the one in Jini [11].
- **Active policy announcement:** Some services such as audio or video tracking might work continuously in the background, without the need for user interaction in order to gather data. In this case, a *privacy beacon* must used that constantly announces the privacy policies of implicitly running data collections, using a short-range wireless link.

2.3 Privacy proxies

Privacy proxies handle privacy relevant interactions between data subjects and data collectors (i.e., policy access and data collection) but also provide access to specific user control capabilities disclosed in the privacy policy such as data updates and deletes, or querying usage logs. Privacy proxies are continuously running services that can be contacted and queried by data subjects anytime, allowing them instant access to their data. Each ubicomp environment either features a single such *service* proxy to handle all its data collections, or multiple service proxies for each individual service it offers. Similarly, each user is expected to have a corresponding *personal* privacy proxy, which handles all interaction between service proxies in order to exchange user data or query their usage logs (in case of disconnects, a mobile device could temporarily act as a substitute for a personal privacy proxy residing on the network). Privacy proxies are configured using a preference language such as APPEL, described above, typically involving a small set of general rules (which could be created by a trusted third party and downloaded by the user) and a larger set of specific rules incrementally created by the user. As part of such an interaction between user and service proxies, an agreement is made in form of an XML-document containing the data elements exchanged and the privacy policy applying to them (both is encoded in the P3P policy). Such an agreement document also contains an explicit *agreement-id* for later reference, as well as detailed information on how the user proxy can access the service proxy (see our extensions to the ACCESS element in figure 2, lines 5-8). Should the user decide to update her email address with all places that have it on file, her privacy proxy contacts each service's update function to transparently update the changed data (*access*).

2.4 Policy-based data access

Once data has been solicited from the user (either actively by receiving a data submission via the privacy proxy, or implicitly by receiving sensor data such as video or audio feed), it is stored in a back-end database (not shown in figure 1 above). In order to prevent accidental use of information that is in disagreement with the previously granted privacy policy, the database not only stores the data collected, but also each individual privacy policy that it was collected under. By combining both data elements and their respective policy into a single unit managed by the database, we can have the database take care of observing that the promises made in a privacy policy with respect to the lifetime, usage, and recipient of a certain piece of information are kept, as well as provide users with a detailed "usage log" of their personal data (*recourse*). Note that since policies are often invariant for a large number of collected data elements, storing an additional pointer to such a policy only adds a small overhead for storage requirements.

3 Implementation

In a first step, two parts of our *pawS* architecture have been implemented: privacy proxies that allow for the automated exchange and update of both privacy policies and user data; and a privacy-aware database (called *pawDB*) that combines the collected data elements and their privacy policies into a single unit for storage in order to consequently handle the data according to its usage policy.

Privacy proxies are implemented as a set of SOAP services running on a Tomcat Apache Web server. Their general method of operation has already been shown in figure 1 above: whenever the user wants to utilize a certain service that requires personal information to be submitted in order to function (e.g., a tracking services that allows telephone calls to be routed to the telephone at my current location), it contacts the service proxy at a URI published either as part of a service discovery mechanisms such as the one in Jini or a continuously running *privacy beacon* (currently simulated). The service proxy replies with a list of available P3P policies (one such policy is shown in figure 2), indicating various levels of service offered and the data needed in each case. Depending on the user's preferences, the user proxy then selects one such policy and replies with the relevant data, using XML messages embedded in SOAP calls. Upon successful completion of the interaction, the service proxy replies with an agreement id that is kept by the user proxy for reference. Depending on each individual agreement, clients can at any time after the data exchange use this agreement id to inspect the personal information stored with the service proxy, or request updates or deletion of their personal data. In the example given in figure 2 (lines 5-8), the service only allows updating the user's password (line 7). While provisions have been made to support digitally signed SOAP messages [3], the current prototype only uses HTTP over SSL to prevent eavesdropping. Authentication is simply done using the agreement id created from the actual data exchange and returned to the client as part of the exchange protocol.

The privacy-aware database, *pawDB*, has been implemented as a Java-API on top of an Oracle 8i database. In a first step, P3P policies describing the reason for the initial data collection (i.e., *data collection policies*) are imported into relational tables using XML-DBMS [1] and are assigned a reference number. Data input through the

API into pawDB then requires not only the actual data elements, but also a link (i.e., the reference number) to such a previously registered P3P policy (policies could also be inserted on-the-fly, this is simply a performance optimization). During insertion, the system compares the submitted data to the privacy policy governing it and transparently stores all data elements together with a link to their privacy policy. In order to query any of the stored data, data users will need to submit a corresponding *data usage policy* themselves (in P3P format), describing in detail who they are, for what purpose they are querying this information, and how long they in turn plan to keep this information. Usage policies are thus not much different from the data collection policies used during the initial data collection. The *pawDB* system then compares each query and its usage policy to the data collection policy of each individual element and transparently withholds a particular piece of information in case of a mismatch between the two. For example, imagine the service provider of a follow-me telephone service offering an improved service and wanting to inform its current users of this. The service provider's marketing division would then draw up a usage policy describing the purpose ("marketing") of the query, as well as its own identity, together with the data elements it needs (e.g., the user's email address). Entering this policy into *pawDB* and running a corresponding query (through the API) referencing this policy, will then return only those email addresses where data owners have consented to marketing purposes. Each such query is recorded in a *data usage log* linked to the agreement id of each recorded data element. This allows data subjects to inspect all usages of their data through the list of recorded usage policies. Furthermore, a daemon process (currently implemented as a simple crontab script) takes care of the guaranteed storage periods set out in the original data collection policies: periodically (e.g., every night), it compares the collection timestamp of a data element to its guaranteed lifetime given in its policy and deletes elements that have been kept longer than the allowed time.

4 Summary and future work

The idea of combining data with metadata governing its use is already popular for enforcing digital copyright [4]. Successful implementation of this concept, however, requires use of so-called "trusted systems" [9] along the whole distribution chain, otherwise it would be fairly easy to separate data and metadata again. In contrast to digital media systems, we are not aiming for hacker-proof data protection but instead assume that the added-value of our trusted system *pawDB* (i.e., having the system make sure that data collector honors privacy policy without costly manual verification) will make its usage popular among data collectors. Of course, it will still be important to add legal requirements to that effect that provide a reasonable recourse mechanism for the few abusers present. *pawS* can also be combined with popular privacy solutions currently developed for the Internet, such as anonymizing tools and identity management systems. Our current privacy proxies can easily be hidden behind anonymizing proxy, such as `anonymizer.com`, thus masking the proxy's identity on the network level and decoupling it from the user's identity. Other popular tools such as Mix-based networks [10] could easily be employed for all wired network communications. If available, *pawS* components could also use anonymizing techniques on the physical layer

as well (e.g., transient MAC-addresses, etc.). Similarly, it should also be possible to incorporate identity management techniques [7] into this framework: every time a data exchange is requested, the user's system can respond with different data set. However, one needs to remember that anonymity and pseudonymity in general might be less useful in a ubicomp environment than on the Internet, simply because real-world data is much more difficult to anonymize completely.

With privacy proxies and *pawDB*, two important components of our *pawS* architecture have been implemented. Our next step is to fully integrate the two components, as well as implementing privacy announcement mechanisms such as privacy beacons or Jini-integrated policy links. Using the P3P extension framework, a mechanism for describing dissemination practices based on the location of the data collection (i.e., the locality and proximity principle from section 2) needs to be incorporated into privacy proxies and *pawDB*. Once a corresponding user interface has been devised, a user-study will finally need to show how useful a tool such as *pawS* will be.

In any case, the scope of *pawS* will remain deliberately limited at providing users of ubicomp environments with a privacy-*enabler*, not with a tamper-proof privacy-*protector*. As we move around in a ubicomp environment, our personal privacy assistant will keep track of all data collections happening with and without our help. Whenever possible, our assistant will enable or disable optional services, based on our preferences. Instead of alerting us to unwanted data collections, however, it might be more useful as a silent but watchful transparency tool keeping track of whom we leave our personal data with. While the actual inspection of its large logs, as well requests for data deletion or updates might be less frequent for the individual user, it is the few cases when we need to know what is going on where it will prove invaluable to us or any consumer interest group, trying to hold data collectors *accountable* to their privacy statements.

References

1. Ronald Bourret. XML-DBMS. Homepage at www.rpbourret.com/xmldbms/.
2. David Brin. *The Transparent Society*. Perseus Books, Reading MA, 1998.
3. Allen Brown, Barbara Fox, Satoshi Hada, Brian LaMacchia, and Hiroshi Maruyama. SOAP security extensions: Digital signature. See www.w3.org/TR/SOAP-dsig, February 2001.
4. ContentGuard, Inc. XrML - the extensible rights markup language. See www.xrml.org.
5. Lorrie Cranor, Marc Langheinrich, and Massimo Marchiori. A P3P preference exchange language 1.0 (APPEL1.0). See www.w3.org/TR/P3P-preferences, April 2002.
6. Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, and Joseph Reagle. The platform for privacy preferences 1.0 (P3P1.0) specification. W3C Recommendation, HTML Version at www.w3.org/TR/P3P/, April 2002.
7. J.J. Borking et al. Intelligent software agents: Turning a privacy threat into a privacy protector. Available at: www.ipc.on.ca/english/pubpres/papers/isat.pdf, April 1999.
8. Marc Langheinrich. Privacy by design - principles of privacy-aware ubiquitous systems. In *Proceedings of Ubicomp*, pages 273–291. Springer LNCS, September 2001.
9. Mark Stefik. Trusted systems. *Scientific American*, pages 78–81, March 1997. Also available online at www.sciam.com/0397issue/0397stefik.html.
10. TU Dresden. JAP - Java Anomyzing Proxy. Homepage at anon.inf.tu-dresden.de.
11. Jim Waldo. The Jini Architecture for Network-centric Computing. *Communications of the ACM*, 42(7):76–82, July 1999.