Obligations: Building a Bridge between Personal and Enterprise Privacy in Pervasive Computing

Susana Alcalde Bagüés[1,2], Jelena Mitic[1], Andreas Zeidler[1],
Marta Tejada[2], Ignacio R. Matias[2], and Carlos Fernandez Valdivielso[2]

[1] Siemens AG, Corporate Technology
Munich, Germany
{susana.alcalde.ext,jelena.mitic,a.zeidler}@siemens.com
[2] Public University of Navarra
Department of Electrical and Electronic Engineering
Navarra, Spain
{tejada.43281,carlos.fernandez,natxo}@unavarra.es

Abstract. In this paper we present a novel architecture for extending the traditional notion of access control to privacy-related data toward a holistic privacy management system. The key elements used are obligations. They constitute a means for controlling the use of private data even after the data was disclosed to some third-party. Today's laws mostly are regulating the conduct of business between an individual and some enterprise. They mainly focus on long-lived and static relationships between a user and a service provider. However, due to the dynamic nature of pervasive computing environments, rather more sophisticated mechanisms than a simple offer/accept-based privacy negotiation are required. Thus, we introduce a privacy architecture which allows a user not only to negotiate the level of privacy needed in a rather automated way but also to track and monitor the whole life-cycle of data once it has been disclosed.

1 Introduction

Over the last few years *privacy topics* have attracted the attention of many researches working in the field of *Pervasive Computing*. The existing common understanding is: the envisioned age of invisible computing is only feasible if people have control over the circumstances under which their personal data is disclosed and how it is processed thereafter. The demand is clear: we should design pervasive computing environments aware of their users' privacy preferences. So far, most efforts are centered around privacy control for enterprises, like E-P3P [1] and EPAL [2]. However, we argue that pervasive computing settings demand an additional level of *personal privacy* complementing enterprise privacy in important aspects. Personal privacy is concerned with maintaining a user's privacy preferences. In our opinion, for guaranteeing an individual's right for privacy, it is necessary to empower a user to decide on the exchange of personal data on a much finer-grained level than possible today. Apart from such mechanisms that provide access control for commercial use, and more recently obligations management [3], users should have their own personalized *context-aware privacy access control*, and additionally the possibility of monitoring the *post-disclosure life-cycle*

S.M. Furnell, S.K. Katsikas, and A. Lioy (Eds.): TrustBus 2008, LNCS 5185, pp. 173–184, 2008.

of the data transmitted. The goal is to enable users to monitor the access, use and deletion of data, also *after* the data was disclosed. Today, this is only possible to the extent that an enterprise "promises" to respect a user's privacy preferences.

We enable post-disclosure monitoring by introducing *obligations* as an independent entity within our User-centric Privacy Framework (UCPF) [4]. *Wikipedia* defines an obligation as *"a requirement to take some course of action"*. In our work presented here, we leverage this notion of an obligation as a required description of regulation on the processing of personal data when being disclosed to third-parties. In this paper, we describe how we add specialized layers for privacy management in order to realize a holistic privacy control, able to fulfill a user's privacy preferences. The key idea is to combine personal and enterprise privacy in an appropriate way. For us it is clear that personal privacy demands differ substantially from those assumed by enterprises, since personal privacy is a much more intimate concern than an enterprise's requirement to meet existing legislations.

This paper is structured as follows: Section 2 compares the requirements for personal privacy protection with those for enterprises. Section 3 then introduces our own privacy framework. Section 4 and 5 are dedicated to our approach for a holistic privacy management based on obligations. The following Sections 6 and 7 are summarizing related work and conclude this paper also indicating directions for future work.

2 Personal and Enterprise Privacy in Pervasive Computing

Pervasive computing scenarios entail the deployment of a large number of *Context-aware Mobile Services* (CAMS) and along with them a "pervasive sensing" of context information related to a person at any time and any place. Therefore, individuals will require automatic mechanisms to control when context is revealed without the need to set their privacy preferences each time and for each service separately. Even the large number of services alone will make a manual per-use authorization of access to personal data (as required by law) an impossible task. Furthermore, individuals will want mechanisms to monitor that enterprises use disclosed data only for fulfilling the requested purpose and nothing else. The challenge here is to meet the individual's expected level of privacy while at the same time dynamic information is revealed in mobile, inherently changing and distributed settings.

Today, enterprises and organizations offer mainly privacy protection mechanisms oriented toward the long-term use of services. They consume personal data, which is classified as *static* in [5], e.g. account number or address. In contrast to the dynamic and short-lived relations typically found in pervasive and mobile settings, where data usually is provided and used only once in a single request. In the latter setting, obviously it is no longer possible to spend the time and effort to define or acknowledge privacy preferences at the moment of use, which is normal for Internet services or company applications. An "offline" solution is needed where a user can define the privacy policy with which a newly discovered service is used; beforehand of actually being in the situation of using it. Our proposal is to add specialized layers of privacy management to give a user a maximum control over the specification and enforcement of his privacy.

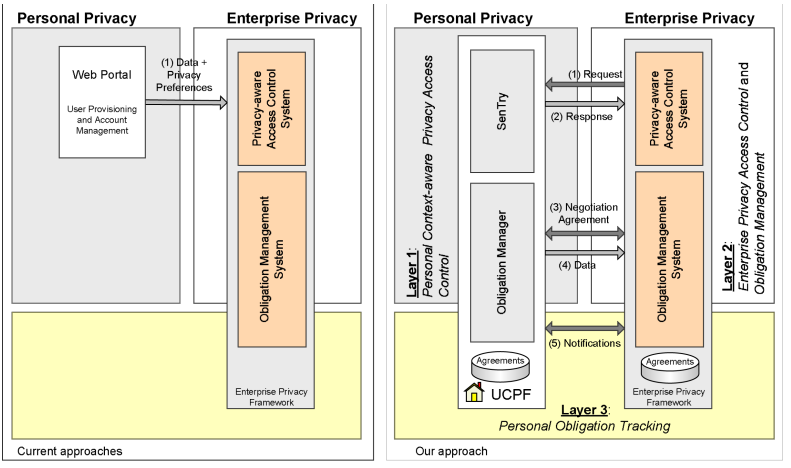

Fig. 1. Privacy Architectures

Figure 1 compares two different situations, on the left hand side the current use of obligations by enterprises, situation applicable to the long-lived service bindings. And on the right hand side our approach of a holistic privacy control scheme integrating access control and obligation tracking on the user side.

In order to address named challenges for privacy in pervasive computing, we have developed a novel privacy architecture consisting of three layers, namely: 1) Personal context-aware privacy access control, 2) Enterprise access control and obligation management, and 3) Personal obligation tracking. These layers are complementary and depend on each other to guarantee a holistic privacy protection. For instance, personal privacy alone cannot protect data once it was transmitted and must rely on enterprise privacy mechanisms. For the rest of the paper we assume that a privacy protection middleware on the enterprise side is in place, capable of handling and enforcing obligations.

The first privacy layer is provided by our privacy framework called UCPF (cf. Section 3). It acts as a personal access filter to delimit when, what, how and who gets permission to access a user's personal data. In pervasive computing scenarios mostly this is related to disclosing context information, e.g. location or activity. Obviously, for users it is desirable to automate such frequent decisions as much as possible and also to have their current context taken into account. For instance, in the example: "Bob allows the disclosure of his location to his employer when his activity state is *working*", the activity of Bob must be checked to decide whether his location is disclosed or not. Therefore, the UCPF's policy system (SenTry) has as design requirement to be *context-aware* [6], in the sense that the evaluation process of a policy might involve consulting a user's context or peer context (e.g. requester) against the applicable policy constraints. We argue that leaving the enforcement of such constraints to a third-party (e.g. enterprise access control system) would not be advisable since it entails the disclosure of sensitive information during the policy evaluation (Bob's activity). Nevertheless, this privacy layer can only address situations where information is disclosed for the present use. But it does not cover cases where information may be stored for future use in a

potentially different context. So, the user has to trust the service to adhere to the legal regulations for enterprises. Here is where the second and third privacy layers are introduced to make users aware of the whole life-cycle of information once it was disclosed.

The second privacy layer is the enterprise privacy access control and obligation management depicted in Figure 1, right hand side. Once data was transmitted, after following the evaluation and enforcement process of the appropriate user's privacy policy in the UCPF, the enterprise service takes over the task of protecting the data. Enterprises are obliged by law to control all accesses to the data gathered from their users. In order to comply with current legislation, enterprise guidelines [7] and individual privacy preferences, enterprise service providers not only should apply traditional access control but also actively accept and enforce privacy obligations from the users. This notion of privacy enforcement is in accordance with the work of Hewlett Packard [3] as part of the European Union project PRIME [8]. Obligations impose conditions for the future that the enterprise is bound to fulfill [9], e.g "My data must be deleted within one month" or "Send a notification when Bob requests Alice location more than N times". This is of vital importance since an enterprise is the only entity in an interaction chain, see Figure 2, able to deal with future situations.

The idea of an enterprise privacy middleware able to enforce obligations based on a user's privacy preferences has been inspired by the work of HP in its Obligation Management System (OMS) [10]. In the OMS framework users can explicitly define their privacy preferences at disclosure time or at any subsequent point of time, e.g., through a web portal. Such privacy preferences are automatically translated into privacy obligations based on a predefined set of templates. As mentioned before this approach is valid for services with long-lived bindings but due to the dynamic nature of CAMS a different solution is needed to automate the exchange of data and privacy preferences. To do so, we impose privacy on enterprises by employing an automatic negotiation protocol over a set of obligations, which contain a user's privacy preferences related with the service used.

The third privacy layer realizes the requirement to empower users of being aware of the "life-cycle" of data after being transmitted. Here, we have developed a set of strategies to reach a trust relationship based on a notification protocol on the agreements stored, at the time of the disclosure between a UCPF and some enterprise service. Agreements include the set of obligations defined by a user in his privacy policy, more details can be found in Section 4.

3 User-Centric Privacy Framework

A first prototype of the UCPF [4] has been developed to be tested on the residential gateway for the Siemens Smart Home Lab. The residential gateway provides access to home-based services from inside and outside the home environment. The incorporation of the UCPF adds privacy control and context brokering as separate functionalities and lets inhabitants interact with outside CAMS. Part of the implementation of the UCPF was incorporated into the privacy framework of the IST project CONNECT [11] as well.

As shown in Figure 2, the UCPF consists of six main functional elements, the SenTry or policy system, the Obligation Manager (OM), the Sentry Registry, the Context

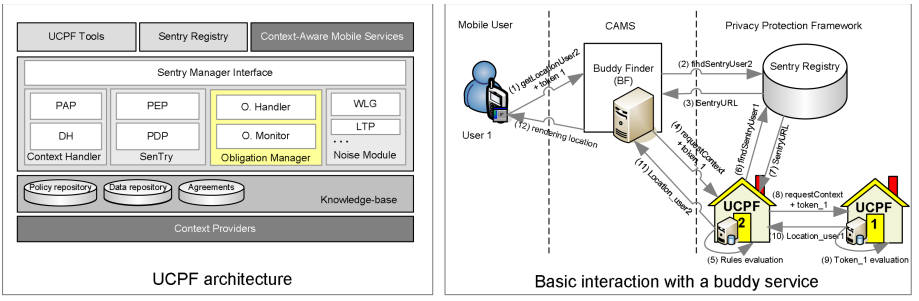

Fig. 2. UCPF overview

Handler (CH), the SenTry Manager Interface (SMI), and the Noise Module (NM). The SenTry is the major architecture building block. It was developed in JAVA on top of the Java Expert System Shell, called Jess. A SenTry instance manages the context disclosure of a user to third parties, based on a set of personal privacy policies defined with the SeT policy language [6]. We benchmarked the performance of the SenTry together with the policy language by using a repository of 500 rules grouped into 6 policies. In average a request took less than 50ms to be evaluated on a standard PC, which seems to be a reasonable performance for the application scenarios considered.

The Obligation Manager, see next Section, negotiates agreements and tracks the obligations agreed on by third parties. The SenTry Registry is the only component that is not co-located with the rest of the elements on the gateway. This component is shared among sentries instances and located in the Internet. It tracks the availability of people's context and provides the pointer to the appropriate SenTry service instance, see Figure 2 right hand side. The interface between SenTry and Service Registry is facilitated by the Context Handler (CH). It supports the identification of external sources of context e.g. for the evaluation of *Foreign Constraints* [12]. Furthermore, the CH acts as a mediator between SenTry and externals context providers. The interaction of end-users with the UCPF is made possible through the Sentry Manager Interface (SMI). It is implemented as an API used to generate, upgrade or delete privacy policies, receive information about the current applicable policies, or getting feedback on potential privacy risks and obligations state. The Noise Module (NM) is a modular component that incorporates additional tools to the policy matching mechanism, e.g. obfuscation and *white lies* [13].

4 Building a Bridge toward Holistic Privacy

In Section 2 the idea of a holistic privacy protection was introduced together with its dependency on the collaboration between a personal privacy framework (the UCPF) and an enterprise privacy system. The question we address now is: *How can this collaboration be established?*, The main problem obviously is that users still have to trust to some degree in enterprises' "promises". Obligations are used to create automatic bindings between both parts, and ensure that data protection requirements are adhered to. However, in cases where those bindings cannot be monitored, checking the compliance

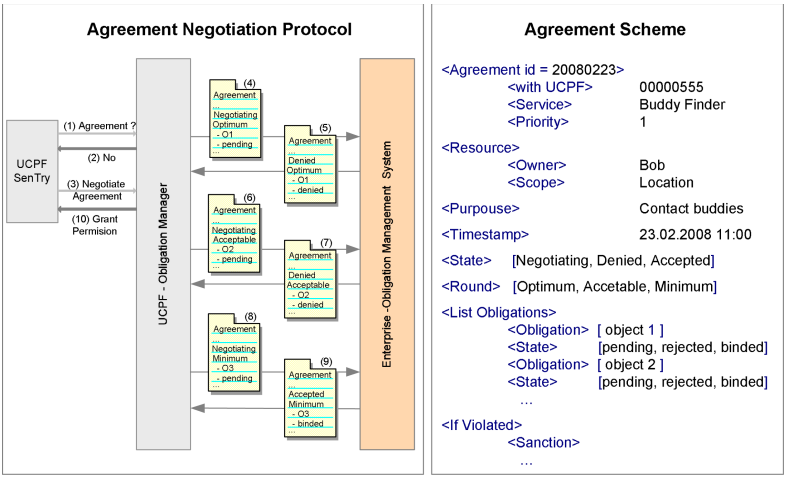

Fig. 3. Agreement Negotiation

with the obligation is almost impossible. The concept of *Non-observable Obligations* is described in the work of Hilty et al. [9], they suggest that a possible solution is the use of nontechnical means, such as audits or legal means. We propose instead the idea of employing *observable* bindings between personal and enterprise frameworks. This is realized by introducing an agreement negotiation protocol together with a trusted notification mechanism, both detailed below.

The agreement negotiation protocol, cf. Fig. 3, starts after the evaluation of a service request within a SenTry instance. If the *rule effect* compiled contains obligations, the *Policy Enforcement Point* (PEP) queries the OM for an agreement over the *pending obligations*, step 3 in Figure 3, and the OM launches the negotiation, steps 4 to 9.

This protocol enables per-service and per-user resource agreements negotiations that are guaranteed to terminate after at most three negotiation rounds. The Obligation Manager makes a first proposal in the *"Negotiating Optimum"* stage. The enterprise side cannot make any counter-proposal at this stage, since the user should not be involved during the negotiation. Therefore, it is limited to check the list of obligations attached and to reject or bind them. If the agreement is denied by the enterprise, which means that one or more obligations are rejected, the OM issues the second proposal: *"Negotiating Acceptable"* stage. It includes a new set of obligations where the rejected obligations of the first set are replaced by their *acceptable* equivalents. The enterprise service may accept the second proposal, or start the third and last round: *"Negotiating Minimum"* stage, in which a new set of obligations classified as *minimum* replaces those rejected. The goals of this negotiation strategy are: i) to allow more than "take or leave" situations, ii) to enable an automatic setup of user's privacy preferences, and iii) to execute the obligation binding process transparent to the user.

In a situation where an enterprise does not accept the third and last proposal, no agreement is reached and the priority of the rejected agreement is taken into account by the OM. Each agreement is labeled with a priority value, *one, two* or *three*. Priority one means that the service (enterprise) MUST accept the agreement otherwise permission

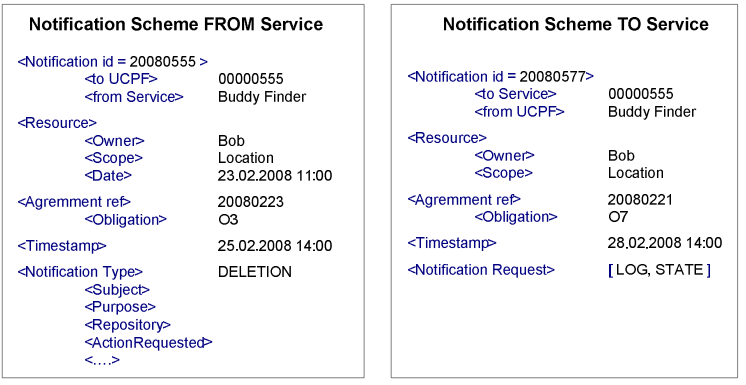

Fig. 4. Notification Schemes

will be denied (step 10). Priority two means that the service SHOULD accept the agreement otherwise data quality will decrease in accuracy. A priority of three means that the service MIGHT accept the agreement and entails the disclosure of the requested data anyway but the user will be notified that no agreement was reached. A user may modify his privacy settings based on the obligations rejected.

Our approach to establish a trusted relationship between an enterprise service and the UCPF is based on the possibility to subscribe to notifications about the use of disclosed data. We introduce two complementary notification types as shown in Fig. 4. The notification template shown on the left hand side is used for notifying the UCPF (as subscriber) about the fulfillment or violation of an obligation by the service provider. We have defined seven notifications types, namely: DELETION, ACCESS, LEAKING, REPOSITORY, REQUEST, DISCLOSURE and POLICY. Depending on the notification type used, further parameters need to be provided. E.g. a DELETION notification does not have any parameter, on the other hand, a DISCLOSURE notification should include at least the *Subject* or *Service*, recipient of the data, and the *Purpose* of such disclosure. The use of notifications allows for monitoring the status of the active obligations and to define actions (penalties) in case of a violation. We introduced the tag *if Violated* (cf. Fig. 3) for this case. It describes the sanctions to be carried out once the OM observes such a violation.

The template on the right hand side of Fig. 4 is the notification scheme used by the UCPF to request a report on the state of or a list of the operations on a particular resource. In summary, notifications are leveraged for: i) enabling monitoring of active obligations, ii) auditing the enterprise service, iii) getting access to personal data in the service's repository (with REPOSITORY notification), iv) knowing when the service's obligation policy changes in order to re-negotiate agreements, and v) controlling when an active obligation is violated.

4.1 Model of Privacy Obligations in the UCPF

In collaboration with the IST project CONNECT, we created a set of 16 obligations as shown in Figure 5. They represent the privacy constraints that a user may impose on

	Description	Action	Notification	Event	System
1	Data MUST not be disclosed to any third-party service	Send Notification	LEAKING	Leaking of data	☑
2	Send notification each time data is disclosed to a subject	Send Notification	DISCLOSURE	Data disclosure	ANP
3	Request permission before any disclosure to a Subject	Send Notification	REQUEST	Data request	ANP
4	Communication must be secured	Encryption		Data transmission	
5	Data in repositories must be encrypted	Encryption		Data storage	
6	Notify the purpose of data access	Send Notification	ACCESS	Data access	ANP
7	Delete data after specified timeout	Delete Data		Timeout	ANP
8	Do not store data in any repository	Delete Data		Session finished	ANP
9	Send notification with URL to the stored data (in service repository)	Send Notification	REPOSITORY	Data storage	☑
10	Send notification when data is removed from repository	Send Notification	DELETION	Data deletion	☑
11	Notify any change of the Obligation Policy	Send Notification	POLICY	Policy changed	☑
12	Send notification when number accesses equals specified value	Send Notification	ACCESS	Data access	ANP
13	Send notification when number disclosures same subject equals specified value	Send Notification	DISCLOSURE	Data disclosure	ANP
14	Send data state when requested by UCPF	Send Data State		UCPF Notification	
15	Send data log when requested by UCPF	Send Data Log		UCPF Notification	
16	Notify change on purpose	Send Notification	ACCESS	Data access	☑

Fig. 5. Obligations defined within the UCPF

an enterprise service when data is disclosed. In our definition, an obligation has two aspects; First, it is a second-class entity subject to the enforcement of a rule by the SenTry component and embedded as part of the rule effect (see Fig. 6, tag *hasEffect*) and to be compiled during the evaluation of a service request. And second, when an evaluation reaches the PEP and it contains obligations, it activates the agreement negotiation protocol as described on Fig. 3. Then, obligations are first-class entities used to convey personal privacy preferences.

In the representation of obligations basically we follow the scheme adopted by the HP framework to facilitate collaboration with the enterprise privacy system. Thus, obligations are XML documents with *Event*, *Action* and *Metadata* elements. Some tags

```
<set:PersonalPolicy rdf:ID="BobPolicy">
  <set:hasTarget rdf:resource="#Bob"/>
  <set:hasRule>
    <set:POR rdf:ID="BobPOR_1">
      <set:onSubject rdf:resource="#Employer"/>
      <set:onResource rdf:resource="#Coordinates"/>
      <set:onAction rdf:resource="#Disclosure"/>
      <set:hasEffect rdf:resource="#BobPOP"/>
      <set:hasConstraint rdf:resource="#BobActivity"/>
    </set:POP>
  </set:hasRule>
  <...>
</set:PersonalPolicy>

<set:POP_result rdf:ID="BobPOP">
   <set:withEvaluation ="grant-with-obligations">
   <set:hasObligation = "BB555">
</set:POP_result>
```

XML document

```
<Obligation id = BB555 >

  <Events>
    <EventType>Timeout</EventType>
    <Period>30</Period>
  </Events>

  <Actions>
    <ActionType>Delete Data</ActionType>
  </Actions>

  <Metadata>
     <Description>Delete data after specified timeout
     </Description>
  </Metadata>

</Obligation>
```

Fig. 6. Obligation's Example

were left out in our definition (e.g. Target), which only can be used together with HP's OMS. In Figure 6 a simple rule example is depicted to show how an XML obligation instance is created to be included in the agreement negotiation protocol (ANP). In this example Bob is allowing his employer to access his location based on the activity, but to delete his coordinates latest after 30 days. The rule is specified using our policy language SeT [6]. The instance of the rule effect (BobPOP) specifies that the result evaluates to "grant" but only if the service agrees on the obligation with id "BB555". Fig. 6, right hand side, shows the XML document referred by the named rule effect.

The table in Fig. 5 shows five special obligations marked as system. Those are mandatory obligations (deducted from current legislation), which are by default established independently of a user's preferences and beforehand of any commercial transaction with a service. The rest marked *ANP* mean that user might include them as a result of the evaluation of a rule and that they will be subject of the negotiation protocol. There are two more obligations highlighted that are obligations that allow the UCPF to audit the enterprise service.

5 Obligation Management from the User Perspective

Due to space restrictions we cannot really go into the details of obligation management. But still we want to give a short introduction to our ongoing work in this area. The question remaining at this point obviously is: *How can a user setup his obligation policies regarding optimal, acceptable, and minimum agreement?*. Figure 7 shows a screenshot of our current application prototype for managing sets of obligations. A user can specify a new rule for being added to his privacy policy and subsequently can allocate a set of obligations to it. A set always consists of the three mandatory obligation types "Optimum", "Acceptable", and "Minimum". These can be predefined and be re-used, obviously, and do not have to be defined separately each time. Implicitly they are always indirectly referenced by id and not by name or privacy rule. For example, in Fig. 6 the id of the obligation chosen is "BB555" which, for the sake of simplicity, is only a single obligation. In our real application the same id would refer to a set of three obligations corresponding to the three mandatory categories.

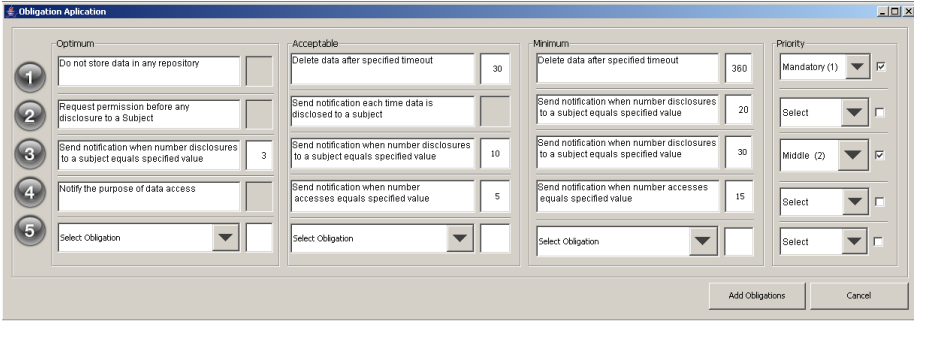

Fig. 7. GUI prototype

In later implementations we hope to use the experiences gathered in field trials to improve the management for the use within different user groups. However, for the time being this has to be considered future work.

6 Related Work

The use of obligations in computer systems by itself is not a new topic. It has been largely used to specify actions that must be performed by some entity. In daily situations where people interact, individuals are held responsible for their own actions; they may be punished if they fail to do what they have promised. In 1996 Van the Riet et al. translated this concept to the "cyberspace". In [14] they conclude that although software entities cannot take real responsibility, their specification still should take into account what such entities must do and what happens if they do not fulfill what has been specified. For instance, within Ponder [15] obligations are event-triggered policies that carry-out management tasks on a set of target objects or on the subject itself, e.g. when a print error event occurs a policy management agent will notify all operators of the error and log the event internally based on an obligation.

In traditional access control systems obligations are coupled tightly to access control policies. An obligation is considered as an action that shall be performed by the system's Policy Enforcement Point (PEP) together with the enforcement of an access control decision. This is the approach followed by EPAL [2], where obligations are entities subordinated to access control. Similarly to EPAL, XACML [16] specifies the syntax and format of access control policies and related obligations. Within this popular access control language obligations are a set of operations associated with an XACML policy that must be executed by the PEP together with an authorization decision. In the work of Park and Sandhu [17] obligations are requirements that have to be fulfilled by a subject at the time of the request for allowing access to a resource. E.g., a user must give his name and email address to download a company's white paper. However, the mentioned approaches do not cover the main requirement of obligations in the context of post-disclosure life-cycle control. Here, obligations should be a binding agreement between two parts, the requesting service and the user, specifying actions to be carried out by the service's PEP after getting the data, at some point in the future. The main goal of these types of privacy obligations is not to constrain the access to the user data but to inform a remote requester of a user's privacy preferences, which should involve an agreement and its posterior tracking.

In Rei [18], an obligation describes an action that must be performed on an object by a distributed entity. An example of an obligation in Rei is "All members of a team are obliged to send weekly reports to the team leader". Rei uses obligations in a similar way to our work and introduces some common and important aspects, such as promises for the future and sanctions in case of violation. However, the Rei framework does not provide an enforcement model. Rei assumes that obligation management is done outside the policy engine although it is not clear how obligations are agreed upon or denied by a third party.

The work presented in [19] describes an approach to archive digital signed commitments on obligations between distributed parties. They introduced the *Obligation*

of Trust (OoT) protocol, which executes two consecutive steps: *Notification of Obligation* and *Signed Acceptance of Obligation*. The OoT is built upon the XACML standard following its Web Services Profile (WS-XACML). The disadvantages of this approach are that it does not cater for the enforcement and monitoring of such obligations, on the one hand, and that it seems to be rather complicated for a common user to manage obligations following this protocol, on the other hand.

We propose a novel privacy architecture in which obligations can be managed by common users. Our framework provides privacy-aware access control, an agreement negotiation protocol over a set of obligations and its posterior tracking. In order to avoid the misuse of private data, once it was disclosed, we rely on the idea of an enterprise privacy middleware able to enforce obligations remotely. This notion of privacy obligations enforcement is in accordance with the work of Hewlett Packard [3] within PRIME [8], as already mentioned in Section 2.

7 Conclusions and Outlook

In this paper we present a novel architecture that extends privacy control in a substantial matter toward holistic privacy management. We introduced the notion of a binding obligation for each privacy-related resource. Such an obligation has to be accepted by a service whenever it requests access to private data. Obligations describe the rights and requirements for processing, storing or deleting data by the service. To avoid situations where obligations are not acceptable by a service and would lead to simple denial of service, we also defined a well-defined negotiation protocol for trying to find an agreement based on different classes of information- or service provided. However, we considered even this to be not far-reaching enough and introduced a third layer of privacy-related functionality: *the personal obligation tracking*, enabling post-disclosure life-cycle awareness. Obligations additionally can describe which information the client (user) wants to receive in order to track the usage of disclosed data after releasing it. We showed that this is an easy but effective way to enable trust between the client and the service. On the other hand, we are aware that we have to gather more experience with these mechanisms. Therefore, we currently are improving the client applications which allow users to maintain and manage their privacy-related settings. This is partly done in the context of the CONNECT project which serves as 'testbed' for the concepts and also provides input for realistic scenarios from different domains.

References

1. Karjoth, G., Schunter, M., Waidner, M.: The platform for enterprise privacy practices - privacy enabled management of customer data. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482. Springer, Heidelberg (2003)
2. Ashley, P., Hada, S., Karjoth, G., Powers, C., Schunter, M.: Enterprise Privacy Authorization Language (EPAL 1.2) Specification (November 2003), `http://www.zurich.ibm.com/security/enterprise-privacy/epal/`
3. Casassa Mont, M., Thyne, R.: A Systemic Approach to Automate Privacy Policy Enforcement in Enterprises. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 118–134. Springer, Heidelberg (2006)

4. Alcalde Bagüés, S., Zeidler, A., Fernandez Valdivielso, C., Matias, I.R.: Sentry@home - leveraging the smart home for privacy in pervasive computing. International Journal of Smart Home 1(2) (2007)
5. Price, B.A., Adam, K., Nuseibeh, B.: Keeping ubiquitous computing to yourself: a practical model for user control of privacy. International Journal of Human-Computer Studies 63, 228–253 (2005)
6. Alcalde Bagüés, S., Zeidler, A., Fernandez Valdivielso, C., Matias, I.R.: Towards personal privacy control. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2007, Part II. LNCS, vol. 4806, pp. 886–895. Springer, Heidelberg (2007)
7. Federal Trade Commission (FTC). Fair information practice principles. Privacy online: A (June 1998)
8. Camenisch, J., et al.: Privacy and Identity Management for Everyone. In: Proceedings of the ACM DIM (2005)
9. Hiltya, M., Basin, D.A., Pretschner, A.: On Obligations. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 98–117. Springer, Heidelberg (2005)
10. Casassa Mont, M.: A System to Handle Privacy Obligations in Enterprises. Thesis (2005)
11. The CONNECT Project, `http://www.ist-connect.eu/`
12. Alcalde Bagüés, S., Zeidler, A., Fernandez Valdivielso, C., Matias, I.R.: A user-centric privacy framework for pervasive environments. In: OTM Workshops (2), pp. 1347–1356 (2006)
13. Alcalde Bagüés, S., Zeidler, A., Fernandez Valdivielso, C., Matias, I.R.: Disappearing for a while - using white lies in pervasive computing. In: Proceedings of the 2007 ACM workshop on Privacy in electronic society (WPES 2007) (2007)
14. van de Riet, R.P., Burg, J.F.M.: Linguistic tools for modelling alter egos in cyberspace: Who is responsible? Journal of Universal Computer Science 2(9), 623–636 (1996)
15. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: Ponder: A language for specifying security and management policies for distributed systems (2000)
16. OASIS standard. eXtensible Access Control Markup Language. Version 2 (February 2005)
17. Park, J., Sandhu, R.: The uconabc usage control model. ACM Trans. Inf. Syst. Secur. 7(1), 128–174 (2004)
18. Kagal, L.: A Policy-Based Approach to Governing Autonomous Behavior in Distributed Environments. Phd Thesis, University of Maryland Baltimore County (September 2004)
19. Mbanaso, U.M., Cooper, G.S., Chadwick, D.W., Anderson, A.: Obligations for privacy and confidentiality in distributed transactions. In: EUC Workshops, pp. 69–81 (2007)