Personal and Ubiquitous Computing manuscript No.
(will be inserted by the editor)

Rafael Accorsi · Matthias Bernauer

On Privacy Evidence for UbiComp Environments

Broadening the Notion of Control to Improve User Acceptance

July 10, 2007

Abstract Today, scepticism prevails on the part of UbiComp users with regard to their privacy. In investigating the reasons behind such a negative mindset, studies often point to the lack of transparency in the collection and usage of personal data: users indeed fear the hidden surveillance through UbiComp technologies. Our thesis is that such scepticism is a consequence of incomplete control mechanisms offered to users in UbiComp environments. Users should not only be *aware* of or capable of *regulating* the collection and usage of personal attributes, but also of *supervising* the system, i.e. be able to obtain credible evidence that the environment does in fact behave as expected. While there is extensive research into regulation, to our knowledge, approaches to tackle supervision are missing. To bridge this gap, we introduce the concept of *privacy evidence* and present the main building blocks towards its realisation.

1 Introduction

The increasing potential to interconnect the computational power of devices with different capabilities given by UbiComp technologies opens up unique chances for novel services and application scenarios, e.g. in the optimisation of logistic processes and, more recently, the provision of individualised customer services.

However, many of these projects were not furthered. In investigating the hurdles to the widespread adoption of UbiComp environments, we are not left with technical issues, as could be initially expected, but rather with sociological issues, namely a lacking acceptance. Driven by pessimistic predictions, such as the "death of privacy" [5] and "privacy as a luxury good" [3], users have a negative attitude when interacting with UbiComp environments. A number of studies substantiate this, showing that users fear UbiComp technologies, such as RFID tags, sensor networks and techniques to correlate multimodal data, as they can be misused for surveillance, thereby leading to the loss of control over their personal data [6,10].

To tackle this problem, research into approaches to mediate the collection of personal attributes and their usage is increasingly gaining in momentum and relevance. Their goal is to allow users (also called "data subjects") to selectively disclose attributes to data consumers, possibly enabling them to formulate policies under which collected attributes can or cannot be accessed or employed. Their rationale is thus to convey a sense of control to users, where "control" stands for the *regulation* of attribute disclosure or, more generally, user exposure.

While approaches to regulate user exposure are indispensable, our thesis is that the sense of control they convey must be broadened. Users today obtain no indication as to whether the systems with which they interact actually behave according to the policies agreed upon, or how the system behaves at all. Put another way, users are left – at best – with a number of privacy promises or expectations, but obtain no credible evidence that their policies have been adhered to. This clearly fails to reproduce the established understanding of control people have in mind, in which control comprises not only the regulation of a set of activities, but also the *supervision* that this set of activities indeed takes place as expected. To our knowledge, approaches on supervision are lacking.

We bridge this gap by investigating the main building blocks necessary to realise supervision in UbiComp environments. In doing so, a conceptional change is due: current techniques aim at an *a priori*, preventive protection of privacy. In contrast, in investigating supervision, we found ourselves in an *a posteriori* setting, where we cannot protect privacy properties, but detect their violation. Both approaches are complementary.

To realise supervision, we employ the concept of *privacy evidence* [10] and elaborate on its realisation. The rationale is to make the behaviour of the UbiComp envi-

Rafael Accorsi
Dept. of Telematics, University of Freiburg.
E-mail: accorsi@iig.uni-freiburg.de

Matthias Bernauer
Inst. for Computer Science, University of Freiburg.
E-mail: bernauer@informatik.uni-freiburg.de

ronment with respect to the compliance to privacy policies evident to users. Intuitively, privacy evidence are records consisting of all the information collected from and related to a particular user – a so-called *log view* – and the result of an automated audit of this log view based on the users' policies. Together, these pieces of information build the basis for supervision and thereby pave the way for a holistic realisation of control.

We firmly believe that investigation towards such a realisation of control is indispensable for privacy in UbiComp. Due to the improved transparency inherent to privacy evidence, supervision has the chance to increase the confidence placed in UbiComp environments and, eventually, even foster the willingness to disclose personal attributes, which is an essential factor for the acceptance of UbiComp technologies in general and the deployment of services and systems in particular. In the long run, both users and system providers could equally profit from such an extended notion of control.

The remainder of this paper is structured as follows. We present an overview of our approach in §2. In §3, we introduce a language for the expression of privacy policies and in §4, log views based on a secure logging service are presented. In §5, we describe our approach to auditing log views based on the stated privacy policies. We briefly report on related work in §6 and discuss our approach in §7.

2 Overview of our Approach

The realisation of privacy evidence anticipates the steps depicted in the schematic workflow in Fig. 1. In detail: a user A formulates a policy P_A and communicates it to the UbiComp environment (1). When interacting with the environment, a number of events are recorded as entries in log files (2). In fact, as we describe below, we assume that *every* event is recorded, so that log files offer a complete digital representation of the activity in a system. At some point in time, possibly after leaving the environment, the user employs a trusted device to retrieve the log view containing all the log entries related to A (3). Based on this, A visualises the collected data and starts a third-party audit process (4) to check whether the policies P_A previously agreed upon have been adhered to, thereby issuing the corresponding privacy evidence (5).

To realise privacy evidence, the following technical building blocks are essential: a *policy language* for the expression of privacy properties; *log views* to allow the visualisation of recorded activity; a *secure logging* to ensure the authenticity of recorded data, in particular to improve the credibility of privacy evidence; and an *automated audit* process for checking the adherence to policies. In the forthcoming sections, we report on these building blocks, thereby putting emphasis on the technical framework necessary to carry out automated audits, a central issue in producing privacy evidence.

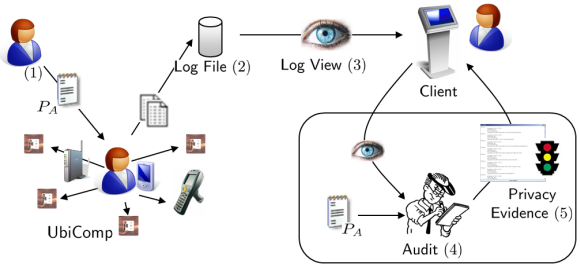

Fig. 1 The workflow for privacy evidence.

Note that the secure storage of log data and the generation of log views are performed on a UbiComp environment that might be untrustworthy. Although we disregard it here, our work also includes a suitable trust model based on trusted computing platforms. The idea is that the daemons responsible for secure logging and log view generation are encapsulated in a trusted sandbox [7], which controls the execution of these processes. Before retrieving the log view, a user (i.e. the process running on his behalf) carries out a remote attestation to ensure that the sandbox, as well as the encapsulated daemons are in place. (This cannot ensure that all the events are communicated to the log daemon; only that the daemon runs safely.) An extended version of this paper elaborates on these technicalities.

Assumptions. We assume that: (a) *every* event happening in the system, as well as every access to collected data is recorded as an event in a log file; (b) in interacting with the system, users are identified while the events they are involved in are recorded. That is, the entries in the log file are always related to a user; and (c) while the system is dynamic in that it adapts itself to the preferences of users, it is static regarding the data collection possibilities. Technically, this means that the ontology describing the system does not change over time and, hence, the policies of users do not become obsolete. Although these assumptions do not hold in general, they hold for some scenarios. We consider the challenges lying beyond these assumptions in §7.

3 On the Expression of Privacy Properties

There are a number of policy languages available today. Prominent examples include, among others, P3P, EPAL and XACML. Although these languages offer more or less the same constructs, they are tailored for specific application scenarios. For example, P3P is employed in the web [12], EPAL within organisations [2] and XACML in grid computing and service-oriented architectures [8].

Instead of sticking to existing policy languages, we rather abstract away from their implementation details and focus on the fundamental notions they are built upon. In doing so, what we propose is an intermediate

```
 1. <Policy>     := (<Rule>) | (<Rule>), <Policy>
 2. <Rule>       := <Col_Ctrl> [, <Cond>] |
                    <Acc_Ctrl> [, <Cond>]
 3. <Cond>       := if (no_prov | <Prov>)
                    and (no_oblig | <Oblig>)
 4. <Col_Ctrl>   := <Perm>, <Subj>, <Obj>, <Event>
 5. <Acc_Ctrl>   := <Perm>, <Subj>, <Obj>, <Right>
 6. <Prov>       := <Atom_Prov> |
                    <Atom_Prov> && <Prov>
 7. <Atom_Prov>  := role <Op> <Role> |
                    purpose <Op> <Purpose> |
                    <Obj> <Op> <Value>
 8. <Oblig>      := <Atom_Oblig> |
                    <Atom_Oblig> && <Oblig>
 9. <Atom_Oblig> := delete <Obj> <Temp_Mod> |
                    notify <Data_Subj> <Temp_Mod>
10. <Perm>       := allow | deny
11. <Right>      := read | write | exec <Cmd>
12. <Temp_Mod>   := within <Nat> days |
                    after <Nat> days
13. <Op>         := > | < | >= | <= | == | !=
```

Fig. 2 Policy language for privacy properties.

policy language placed between the algebraic semantic foundation of languages and their realisations.

Two notions build the basis of our language: *access* to pieces of information and their *collection*. (We regard this distinction as essential, as UbiComp environments enable implicit interactions with users [11], which must also be controlled.) Atomic access and collection rules may not be expressive enough in a number of situations though. For example, a user may categorically prohibit the collection of RFID information, but allow it if he is notified about this collection.

This kind of condition is referred to as *usage control*. Usage control extends traditional access control techniques by allowing users to formulate *provisions* and *obligations* [9]. Provisions express the conditions that must be fulfilled in order to grant or deny access or collection. Obligations express events that must occur once access or collection is granted or denied. (Obligations are equipped with sanctions denoting compensations in case of non-adherence. Sanctions are often enforced by means other than an execution monitor and therefore omitted here.) Based on this, Def. 1 sets out the core language $\mathcal{P}$.

Definition 1 The policy language $\mathcal{P}$ is defined by the BNF in Fig 2. Senteces of $\mathcal{P}$ represent policies P and the rules of P are denoted by r. Let $\mathtt{e}$ be a nonterminal, $\psi(r, \mathtt{e})$ is the terminal associated to $\mathtt{e}$ in r. □

In Fig. 2, items enclosed in square brackets denote optional constructs. A policy is a finite sequence of rules $P = r_1, \ldots, r_n$ (Prod. 1). Without loss of generality, we assume below that P is a *set* of rules. Rules are classified according to their structure as:

- *atomic* rules have neither provisions nor obligations.
- *conditional* rules encompass provisions or obligations.
- *provisional* rules have only provisions.
- *obligational* rules comprise only obligations.
- *full* rules encompass both provisions and obligations.

Each rule is further distinguished according to the notion it takes into account, i.e. access or collection. Unless stated otherwise, the term "rule" applies to both notions.

In formulating atomic collection rules (Prod. 4), the user stipulates whether a certain subject is able to collect an attribute and/or through which events this can occur. In formulating atomic access rules (Prod. 5), the user stipulates the rights (Prod. 11) of subjects over particular attributes. In our realisation, the wildcard `*` can be used in both productions to represent a whole class of items, e.g. subjects or attributes. Provisions (Prod. 7) regard the role a subject engages in, as well as the purpose of the access or collection and the value of collected data fields (objects) serving as guards to an access or a collection; obligations (Prod. 8) encompass the deletion of an attribute according to some temporal modality (Prod. 12) and the notification of individuals.

Example 1 User A stipulates that: (1) RFID collection is prohibited; and (2) the transactions of A involving a sum lesser than $100 can be read by the accounting department for statistic purposes, whereas the access must be notified within a week. The policy P_A is r_1, r_2 where r_1 and r_2 stand respectively for:

```
(deny, RFID-Reader, *, *)  and
(allow, *, Trans.Value, read,
  if (Trans.Value < $100 &&
      role == Account     &&
      purpose == Statistic)
  and (notify A within 7 days)).  □
```

Formally, policies characterise a finite set of discretionary authorisation rules denoting safety properties.

4 Secure Logging and Log Views

Log data is a central source of information in considering privacy evidence in particular and approaches to checking compliance to rules in general. In contrast to "static" files, such as text documents or spreadsheets, log files allow one to reconstruct the dynamics of a system, i.e. the course of events that led to some particular state.

To be credible, log data must exhibit *integrity*, i.e. log data is accurate (entries have not been modified), complete (entries have not been deleted) and compact (entries have not been illegally added to the log file), and be *confidential* in that log entries cannot be stored in clear-text, for such log data can be easily duplicated. Moreover, techniques to ensure these properties need to account for *tamper evidence*, i.e. attempts to illicitly manipulate log data are detectable to a verifier, and *forward integrity*, i.e. if an attacker succeeds in breaking in at time t, log data stored before t cannot be compromised.

Based on [1], we summarise the realisation of a secure logging service that fulfils these requirements. Assuming

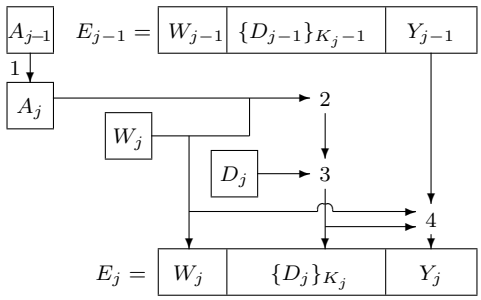

Fig. 3 Appending an entry to the log file.

that each application carries out its own logging, we consider an additional log file BBox that can be seen as a black box of the system. It receives events from, e.g., transactional and operational log files of databases, decisions of reference monitors and other sensing devices. The communication between these devices and the collectors are secured using asymmetric cryptography. At receiving a log message, each log event D_j is (symmetrically) encrypted with an evolving cryptographic key K_j obtained from a secret master key A_j and an index field W_j. (W is used to describe the user to whom the entry refers.) A hash chain Y associates the previous entry E_{j-1} and E_j. This procedure is depicted in Fig. 3, where the numbers correspond to:

1. $A_j = \mathit{Hash}(A_{j-1})$ denotes the authentication key of the jth log entry. The confidentiality of A is essential as it is used to encrypt log entries. Thus, we assume that the computation of the new value irretrievably overwrites the previous value, and that A_0 is stored in a safe manner, possibly off-line.
2. $K_j = \mathit{Hash}(W_j, A_j)$ is the cryptographic key with which the jth log entry is encrypted. This key is based on the index W_j, so that only corresponding user gains access to an entry.
3. $\{D_j\}_{K_j}$ is the encrypted log entry D_j.
4. $Y_j = \mathit{Hash}(Y_{j-1}, \{D_j\}_{K_j}, W_j)$ is the jth value of the hash chain. Each link of the hash chain is based on the corresponding encrypted value of the log data. This ensures that the chain can be verified without the knowledge of the actual log entry D_j.

The resultant log entry $E_j = W_j, \{D_j\}_{K_j}, Y_j$ consists of the index W_j, the encrypted log entry $\{D_j\}_{K_j}$ and the hash chain value Y_j. Considering the generation of privacy evidence, we define the form of a logged event D.

Definition 2 A log event D is defined by the BNF in Fig. 4. Sets of log entries are denoted by S. □

An entry thus consists of a timestamped identification recording the action of a subject (Prod. 1). We consider three types of actions (Prod. 2), namely collection, access and usage, each of which is equipped with the necessary details. Note that entries may be related to others,

```
1. <Entry> := (<Id>, <Time>, <Subj>, <Type>)
2. <Type>  := Col, <Event>, <Value>, <Ctrl> |
              Acc, <Right>, <Ctrl> |
              Usa, <Ref_To>, <Action>
3. <Ctrl>  := <Ref_To>, <Role>, <Purpose>,
              <Obj>, <Perm>
```

Fig. 4 Form of logged events.

e.g. in recording an obligation, the entry refers to a previous entry that triggered the obligation. Such relationships are denoted here by the field <Ref_To>. The following constructs are used later in generating evidence.

Definition 3 Let D be a log event and e a nonterminal expression. $\pi(D, \mathtt{e})$ defines the value in D associated with the terminal expression. Let i be an index such that $i = \pi(D, \mathtt{Ref\_To})$. $\rho(D) = D'$ such that $\pi(D', \mathtt{Id}) = i$. □

The projection π returns the value of a field in an entry D, while $\rho(D)$ the entry to which D refers to.

Log Views. A central concept to enable supervision is to give users timestamped information regarding *which* attributes have been collected, *who* has had access to them and *how* these attributes have been used. In our approach, these pieces of information are compiled into *log views* L [10], a concept bearing similarity with its homonymous counterpart in databases. Log views $L = (S, M)$ consist of an individualised audit trail S comprising factual data (performed transactions, collected attributes) and monitored data (access and usage information) about a user, and meta data M about the generating UbiComp environment and the view's integrity.

To retrieve a log view L, user A employs a trusted device (e.g. a personal computer or a terminal dedicated to this purpose) to authenticate himself to the UbiComp environment, which then starts a query over (possibly distributed) BBoxes. To this end, the index W of each entry is checked against the authenticated user. If they match and the entry passes an integrity check based on the hash chain, the content D of the entry is decrypted and appended to the corresponding S. When all the entries are queried, the log view is signed and sent to A.

5 Compliance Audits and Privacy Evidence

Log views would suffice, at least in theory, to realise the holistic sense of control we argue for in this paper: users could browse through the entries and check whether their privacy policies have been adhered to or not. But this is easier said than done: log views include even in small experiments thousands of entries and their interrelationships are unclear and hard to reconstruct, regardless of how much effort we put into improving their readability.

To tackle this problem, we develop an approach for automated compliance audits to check the adherence to

policies. Given a policy P and a log view $L = (S, M)$, we first define a transformation ν that takes P and returns the set of rules $V = \{v_1, \ldots, v_n\}$, such that each $v_i \in V$ denotes the violation of a rule in P. Each violation v_i is then checked against the audit trail S.

To define ν, we first define in Def. 4 the polarity of a rule and in Def. 5 negation of provisions and obligations.

Definition 4 Let P be a production of $\mathcal{P}$. $r \in P$ has a positive polarity $(+r)$ if $\psi(r, \texttt{Perm}) = \texttt{allow}$; r has a negative polarity $(-r)$ if $\psi(r, \texttt{Perm}) = \texttt{deny}$. The function *pol* returns the polarity of r and *invpol* reverts the polarity of r, i.e. $invpol(r) = -r$ if $pol(r) = +$ and $invpol(r) = +r$ if $pol(r) = -$. □

Definition 5 Let p denote a sentence of `<Atom_Prov>` and t a sentence of the temporal modality `<Temp_Mod>` of $\mathcal{P}$. We define $p^{\perp}$ and $t^{\perp}$ as follows:

- if $p =$ `role <Op> <Role>`, $p^{\perp} =$ `role <Op'> <Role>`
- if $p =$ `purpose <Op> <Purpose>`, $p^{\perp} =$ `purpose <Op'> <Purpose>`
- if $p =$ `<Obj><Op><Value>`, $p^{\perp} =$ `<Obj><Op'><Value>`
- if $t =$ `within <Nat> days`, $t^{\perp} =$ `after <Nat> days`
- if $t =$ `after <Nat> days`, $t^{\perp} =$ `within <Nat> days`

Here, `<Op'>` stands for the logical negation of the corresponding binary operator `<Op>`. Let o be a production of `<Atom_Oblig>`, $o^{\perp}$ is obtained by replacing the temporal modality t of o by $t^{\perp}$. □

Definition 6 Let P be a sentence of $\mathcal{P}$. The transformation $\nu(r)$ is defined as:

1. Let r be an atomic rule, $\nu(r) = v = invpol(r)$.
2. Let r be a provisional rule with n provisions $p_1, \ldots p_n$. $\nu(r) = v_1, \ldots, v_n, v_{n+1}$ such that each v_i for $1 \leq i \leq n$ stands for the rule r with the ith provision p_i replaced by $p_i^{\perp}$ and $v_{n+1} = invpol(r)$.
3. Let r be an obligational rule consisting of m obligations $o_1, \ldots, o_m$. $\nu(r) = v_1, \ldots, v_m, v_{m+1}$ v_i with $1 \leq i \leq m$ stands for the rule r with the ith obligation o_i replaced by $o_i^{\perp}$ and $v_{m+1} = invpol(r)$.
4. Let r be a full rule with n provisions $p_1, \ldots, p_n$ and m obligations $o_1, \ldots, o_m$. $\nu(r) = v_1, \ldots, v_n$, $v_{n+1} \ldots v_{n+m}, v_{n+m+1}$, where each i such that $1 \leq i \leq n$ is obtained as in step 2, each j such that $n+1 \leq j \leq n+m$ is obtained as in step 3 and $v_{n+m+1} = invpol(r)$.

We define ν over a set of rules $P = \{r_1, \ldots, r_n\}$, such that $\nu(P) = V$ where $V = \{\nu(r_1), \ldots, \nu(r_n)\}$. □

According to Def. 6, the violation of an atomic rule r is obtained by reverting the polarity of r. Violations of conditional rules are obtained by negating their conditions, where we also consider the case where conditions are fulfilled and the actual access or collection is not adhered to. Let us consider the rules in Example 1. The $\nu(r_1)$ is denoted by

```
(allow, RFID-Reader, *, *)
```

and one of the violations v of $\nu(r_2)$ is denoted by

```
(allow, *, Trans.Value, read,
 if (Trans.Value < $100 &&
     role != Account &&
     purpose == Statistic)
 and (notify A within 7 days)).
```

Note that the resultant violations are always rules in $\mathcal{P}$.

With ν at hand, we now search S for violations in a manner similar to model-checking, except that we do not have a formal model of the system, but the resultant execution traces, a subset of which is S. To this end, we define the pinpoint relation $\triangleright$ between the audit trails of a view and the set of violations V: $S \triangleright V$ iff $v_i \in V$ can be pinpointed, i.e. detected, in S. If there is a $v_i \in V$ such that $S \triangleright V$, then there is an execution of the system that violates r_i and, in consequence, the policy P.

Definition 7 Let $L = (S, M)$ be a log view, P a policy and $V = \nu(P)$. $S \triangleright V$ iff $\exists D \in S$ and $\exists v_i \in V$ such that:

1. Let v_i be generated by an atomic rule. For all nonterminals `e` of v_i, $\psi(v_i, \texttt{e}) = \pi(D, \texttt{e})$.
2. Let v_i be generated by a provisional rule.
   (a) Role or purpose provision: for all nonterminals `e` of v_i, $\psi(v_i, \texttt{e}) = \pi(D, \texttt{e})$ and
   (b) Guarded provision: $\psi(v_i, \texttt{Obj}) = \pi(\rho(D), \texttt{Obj})$ and $\psi(v_i, \texttt{Value})\ \texttt{op}\ \pi(\rho(D), \texttt{Value}) =$ True, where `op` stands for the operator $\psi(v_i, \texttt{Op'})$.
3. Let v_i be generated by an obligational rule. For all nonterminals `e` of v_i with the exception of `Action`, $\psi(v_i, \texttt{e}) = \pi(D, \texttt{e})$, and if $\exists D' \in S : \rho(D') = D$, such that $\psi(v_i, \texttt{Action}) = \pi(D', \texttt{Action})$, then
   (a) if $\psi(v_i, \texttt{Temp\_Mod}) = \texttt{within}$ then $\pi(D', \texttt{Time}) \leq \pi(D, \texttt{Time}) + \psi(v_i, \texttt{Nat})$
   (b) if $\psi(v_i, \texttt{Temp\_Mod}) = \texttt{after}$ then $\pi(D', \texttt{Time}) > \pi(D, \texttt{Time}) + \psi(v_i, \texttt{Nat})$. □

In Def. 7, we distinguish between the kinds of rules. For atomic rules, as well as provisional rules involving roles and purpose, we pattern match the violation with the entries. For guarded provisions, we evaluate whether the guard matches with the accessed values. For obligations, we first need to verify whether the action (delete and notification) has taken place in the first place. If so, we check whether the temporal modality has been violated. The item `*` is, as a matter of simplicity, not included in our definition. If a violation contains a `*`, the pattern matching always evaluates to true.

Privacy Evidence. The result of the compliance audit, together with the corresponding log view, build the privacy evidence, where we employ a semaphore notation to make the result of audit evident to the pertinent user. Here, red denotes the *violation* of some rule, while green denotes the *compliance* with a policy. An amber semaphore indicates that some obligational rule could not be pinpointed and therefore stands for a *warning*.

Formally, this is special case of Def. 7 omitted above for simplicity. It refers to the situation where an action is due, yet its deadline has not expired. Put other way, the audit happens at some time point t between the triggering collection or access and the expiration deadline.

Besides obtaining privacy evidence, the user can further interact with it by clicking over the semaphore to obtain details on which rules have been violated or warned – if any –, as well as the specific entries that led to this result. We currently carry out experiments to examine the overhead involved in generating privacy evidence.

6 Related Work

Compliance audits is a young research topic and we are not aware of research into privacy evidence in the sense we propose here. Nevertheless, our approach is tightly related to EPAL, the enterprise privacy authorization language introduced by the IBM [2]. There are, however, several differences. First, the primary focus of EPAL is the enforcement of policies. In contrast, our focus is the a posteriori inspection of compliance with privacy policies. Second, EPAL is enterprise-centric, i.e. the notion of user or data subject is not emphasised. Privacy evidence focus on the users and on broadening their sense of control. Third, in EPAL it is not clear who is in charge of auditing the system. In our approach, a third-party process acting on the behalf of the user assumes the auditor role.

A posteriori policy compliance is studied in [4], where a formal proof system and its mechanisation are presented. While the formal background is sound, the practical contribution of the approach and the operational mode (who performs the audit and how) are unclear.

7 Discussion and Perspectives

We argue for a holistic notion of control for UbiComp comprising the *regulation* of collection of and access to attributes and the *supervision* of compliance to policies. To this end, we introduce privacy evidence as a means of provable enforcement and present some of its main building blocks. In doing so, we aim to improve the acceptance of UbiComp and foster the willingness of users to use the services these environments can provide.

We believe privacy evidence is an initial step in a promising direction and many relevant issues remain to be investigated. Our realisation of privacy evidence currently works under a strong set of assumptions, which while holding in the scenarios we consider, cannot be anticipated in general. One open question concerns the guarantee that relevant events are sent to the log file BBox used to generate privacy evidence. A possibility is to extend the trusted computing base to encompass the devices, so that they ensure that every event communicated to the local log file is also send to the BBox.

Moreover, the assumption that users are recognisable when an event is recorded may be at odds with their desire to remain anonymous. This need not to be the case though, as an identification may also be related to a user's pseudonym. (Note that this possibility depends on the requirements set by the scenario.)

Another open question concerns situations where log entries cannot be assigned to a particular user. To address this issue, we plan to equip the generation of log views with data mining and machine learning methods: given a log view L of a user, these methods anticipate whether, and, if so, to what extent an event D is related to the set of log entries S. Our present investigation considers cluster building techniques for this purpose.

Finally, if the ontology describing the system changes, the policies become (partially) obsolete and events captured by the new devices will not be taken into account in the compliance audit. In fact, this generates a semantic problem that can be resolved using default configurations, such as default-deny. In this case, unknown events would be seen as violations or warnings.

Thus, in relaxing our assumptions, we consider a setting that resembles a forensic investigation. Our medium-term goal is indeed to argue for the establishment of *privacy forensics*, a discipline dedicated to the technical and legal research into methods for compliance audits and privacy evidence mining, protection and presentation.

References

1. R. Accorsi. On the relationship of privacy and secure remote logging in dynamic systems. In S. Fischer-Hübner et al., *IFIP/SEC*, vol. 201 of *IFIP*, pp. 329–339, 2006.
2. P. Ashley, C. Powers, and M. Schunter. From privacy promises to privacy management: A new approach for enforcing privacy throughout an enterprise. In *ACM NSP*, pp. 43–50, 2002.
3. R. Böhme and S. Koble. On the viability of privacy-enhancing in a self-regulated business-to-consumer market: Will privacy remain a luxury good? In *WEIS*, 2007.
4. J. Cederquist, R. Corin, M. Dekker, S. Etalle, J. den Hartog, and G. Lenzini. Audit-based compliance control. *Int. J. of Information Security*, 6(2-3):133–151, 2007.
5. M. Froomkin. The death of privacy? *Stanford Law Review*, 52(5):1461–1543, May 2000.
6. O. Günther and S. Spiekermann. RFID and the perception of control: The consumer's view. *CACM*, 48(9):73–76, 2005.
7. A. Hohl. *Traceable Processing of Personal Data in Remote Services Using TCG*. PhD thesis, Freiburg, 2006.
8. OASIS. Extensible access control markup language. `http://www.oasis-open.org/committees/xacml/`.
9. A. Pretschner, M. Hilty, and D. Basin. Distributed usage control. *CACM*, 49(9):39–44, 2006.
10. S. Sackmann, J. Strüker, and R. Accorsi. Personalization in privacy-aware highly dynamic systems. *CACM*, 49(9):32–38, 2006.
11. A. Schmidt. Implicit human computer interaction through context. *Personal and Ubiquitous Computing*, 4(2-3):191–199, 2000.
12. W3C. P3P, `http://www.w3.org/P3P/`.