Protecting Privacy

No Author Given

No Institute Given

Abstract. Context-awareness and privacy protection is a pair of contradictory requirements in context-aware computing. On one hand, context-aware systems must collect sufficient context information reflecting physical surroundings such as users' location, activity, habits, etc. to make intelligent decisions without user interaction. On the other hand, context is often linked to individuals (e.g., location of a person) and as such falls under privacy directives. To reconcile privacy protection (automatically fulfilling users' privacy whishes) and context-awareness (provided by context histories), This chapter first reviews privacy protection techniques in the database field. Two context privacy protection strategies, including encrypted context information searching and life-cycle management of privacy-sensitive context information, are particularly detailed.

1 Balancing Privacy and Smartness

Undeniably, computing systems are more and more aware of their surroundings, thanks to widely spread smart devices able to continuously nourish the computing infrastructure with information describing the real world [?]. Not only lightweight devices like cell phones, PDA, RFID tags localizing objects [?, ?] and employees but also chips penetrate commonly used objects such as clothes [?], televisions [?], GPS-equipped cars, floors [?], etc., and even target at the human body [?, ?]. This new combination of technologies is pushing towards so called pervasive computing having real-time knowledge of the surroundings, and this knowledge is generally referred to as context.

Context gives the ability to computing systems to become context-awareness, i.e., endowing the systems with enough smartness to (1) minimize user-machine interaction, and (2) be deployed in daily life areas (like streets, supermarkets, homes, etc.) with non-traditional stationary desktop-based computing interfaces. Continuously sensed context, reflecting individuals' location, behavior, mood, or habits etc., is needed to reach those smartness requirements, i.e., to adapt applications to fit the environment; infer information or draw conclusions from rules and observations; learn in the sense of using experiences to improve performance, and to anticipate what to do next [?]. However, an obvious but important remark is that the smartness of context-aware systems is tightly coupled to accuracy quality and quantity of the available past and present context, as exposed in [?, ?] dedicated to usage of context histories in smart environments.

Although context information might be considered less sensible than traditional data like health care folders or banking information, it falls under privacy regulation when linked to an individual (e.g., location of a person), given the definition of *personal data* in the worldwide privacy acts [?]. Particularly, with context-aware ambient intelligent spaces being planned to be developed everywhere, this will lead in the extreme to sense and archive everyone's actions and moves everyday [?,?]. Imagine how many of us would feel comfortable if every of our activities was sensed and tracked by our surroundings. Also, with sufficient accuracy and volumes, context may become of main interest [?] for malicious usage, e.g., to monitor employees in companies, detect behavior of inhabitants and/or company staff to prepare a robbery or for marketing purposes, or check people behaviors before contracting insurance contracts. This inevitably constitutes a critical privacy threat.

Nevertheless, Enforcing privacy protection in context-aware applications generates the difficulty in controlling and exploiting the content (volume and accuracy) of context histories. In fact, privacy has widely been recognized as being application killer in context-aware ambient intelligent computing, forming a crucial problem. The goal of this chapter is to investigate this issue and provide sensible solutions to reconcile context-awareness and privacy in the ambient intelligent world.

2 Privacy Protection Techniques

Privacy is the right of individuals to determine for themselves when, how, and to what extent information about them is collected, stored, and communicated to others. In the data management field, several research efforts have been devoted to address issues related to the development of privacy preserving data management techniques.

2.1 Access Control

Overview of Access Control The function of access control is to control which principals (persons, processes, machines, etc.) have access to which resources in the system [?]. It is the traditional center of gravity of computer security. In the database field, access control is widely used to exert control who can get access to a subset of the database where much sensitive structured data resides. The current SQL standard allows coarse-grained access both to database tables as well as views. Fine-grained access control policies based on authorization predicates are also enforced in [?,?,?]. For instance, each employee in an organization is authorized to access his/her own record in the employee table. Typical access control models include discretionary access control, mandatory access control, role-based access control, and purpose-based access control.

I. Discretionary Access Control

Discretionary access control (DAC) is an access policy determined by the owner of the resource. The owner decides who is allowed to access the resource and what privileges he/she has [?]. The resource's initial owner is usually the subject that caused it to be created. DAC is the traditional file access restriction mechanism in unix systems.

II. Mandatory Access Control

Mandatory access control (MAC) is an access policy, allowing access to the resource if and only if corresponding rules exist that allow a specific user to access the resource [?]. In MAC, sensitive resource is usually protected using a partial ordering of sensitivity levels (e.g., top-secret, confidential, classified, etc.). Every user and resource have sensitivity labels assigned to them. A user's sensitivity label specifies its level of trust. A resource's sensitivity label specifies the level of trust required for access. In order to access a given resource, the user must have a sensitivity level equal to or higher than the requested resource. Two methods are commonly used for applying mandatory access control. (1) Rule-based (or label-based) access control, which defines specific conditions for access to a requested resource. (2) Lattice-based access control, which defines complex access control involving multiple resources and/or users.

III. Role-based Access Control

Role-based access control (RBAC) is an access policy determined by the system, not the owner [?, ?]. Unlike DAC which allows users to control access to their resources, RBAC controls resource access at the system level outside of the user's control. It defines which permissions belong to which role. A role in RBAC can thus be viewed as a set of permissions. Roles can then be assigned to users. This makes administration of access to sensitive resources easier: a particular user is simply assigned an appropriate role, and this role defines the permissions. Roles can be combined in a hierarchy where higher-level roles subsume permissions owned by sub-roles.

IV. Purposed-based Access Control

Purposed-based access control (PBAC) is an access policy, which associates purpose information to the resources [?]. It regulates access to those resources based on the purpose for which it needs to be accessed. By using the concept of intended purposes, it is possible to describe for which purposes the resources can be accessed, and which purposes cannot be used to access the resources. It is the system's responsibility to determine the access purpose, and match this with the set of intended purposes to decide whether or not access will be granted. Access purposes can be associated with roles, which can be managed using regular RBAC techniques.

Combining Access Control and Privacy Protection Access control can be employed to restrict access to and use of data, thereby, limit information disclosure according to privacy policies. Recently, Chaudhuri *et al.* present an architecture to integrate access control primitives and privacy preserving mechanisms within a database system in a principled manner [?]. The basic idea is to enhance an authorization policy with the abstraction of *noisy* views that encapsulate previously proposed privacy mechanisms. These noisy views represent authorization views and are implemented based on differentially private algorithms. By combining authorizations and differentially private views in this manner, queries that refer to both the base tables and the differentially private views can be supported, resulting in a system that is more powerful than using access control techniques or differential privacy techniques in isolation. On a cloud setting, Roy *et al.* also combine access control primitives with differential privacy, where the execution engine is MapReduce [**?**].

2.2 Platform for Privacy Preferences (P3P)

To give users control of their personal information when browsing web sites, the World Wide Web Consortium (W3C) develops the Platform for Privacy Preferences (P3P). It allows Web sites to declare their intended use of information they collect about web browser users in a machine-readable XML format, known as P3P policy, which can programmatically be compared against user's privacy preferences [**?**].

Overview of P3P The basic mechanism of P3P is as follows. When a web site uses P3P, it sets up a set of policies that allow it to state the intended uses of personal information that may be gathered from the site visitors. When a user decides to use P3P, s/he sets the own set of policies and states what personal information s/he allows to be seen by the web sites that they visit. Then when a user visits a site, P3P will compare what personal information the user is willing to release, and what information the server wants to get. If the two do not match, P3P will inform the user and ask if he/she is willing to proceed to the site, As an example, a user may store in the browser preferences that information about their browsing habits should not be collected. If the policy of a web site states that a cookie is used for this purpose, the browser automatically rejects the cookie [**?**].

The P3P protocol is comprised of two parts.

(1) *Privacy policies* indicate which information will be collected? for what purpose? who may see the information? and for how long the information will be kept. They are described in an XML format as a sequence of elements.

(2) *Privacy preferences* are a machine-readable specification of user's preferences that can be programmatically compared against a privacy policy. They are expressed as a list of rules. Each rule includes a rule behavior and a rule body. The rule behavior specifies the action to be taken if the rule fires. The behavior can be request, implying that the policy conforms to preferences specified in the rule body. It can be block, implying that the policy does not respect user's preferences. See [**?**] for other behaviors. The rule body provides the pattern that is matched against a policy. The format of a pattern follows the format used in specifying privacy policies.

Limitations of P3P Although P3P enables users to have control over the information a web site collects, it does not specify any mechanisms for enforcing that sites act according to their stated privacy policies [?]. P3P policies published by web sites are thus not trusted by users. The resulting P3P framework does not provide a coherent view of available privacy protection mechanisms to the user [?]. Besides, low P3P adoption impedes client adoption by users. The languages available to describe user privacy preferences are also not sufficiently expressive.

Implementing P3P in Databases and Enterprises Agrawal *et al.* apply the database technology to checking P3P privacy policies against users' privacy preferences at the server side [?]. A web site deploying P3P first installs its privacy policies in a database system. User's privacy preferences are translated into SQL or XQuery. Then database querying technology is used for matching user's preferences against privacy policies. The performance study [?] shows that it can lead to adequate performance in practical deployment of P3P [?].

Similar to web sites, enterprises often collect large amounts of personal data from their customers. To enforce P3P-like policies in enterprises, Ashley *et al.* outline an architecture based on access control techniques [?], where enterprises publish privacy statements that outline how data is used and shared. The Platform for Enterprise Privacy Practices, referred to as E-P3P, is then defined as a fine-grained privacy policy model. A chief privacy officer uses E-P3P to formalize the internal handling of collected data in the enterprise. A user is allowed to use certain collected data for a given purpose if and only if the E-P3P authorization engine allows this request based on the applicable E-P3P policy. By enforcing such privacy practices, E-P3P ensures enterprises to keep their promises and prevent accidental privacy violations.

2.3 Hippocratic Databases

The Hippocratic Oath has guided the conduct of physicians for centuries, serving as the basis of doctor-patient relationship.

"*And about whatever I may see or hear in treatment, or even without treatment, in the life of human beings C things that should not ever be blurted out outside C I will remain silent, holding such things to be unutterable*" - Hippocratic Oath

In the same spirit as the Hippocratic Oath by doctorsthey swear to ethically practice medicine, Agrawal *et al.* introduce the concept of Hippocratic databases to urge the database systems to respect and enforce the privacy once users' privacy-sensitive data enters the systems [?]. Hippocratic databases are built upon the following ten principles, derived from real-world privacy regulations and guidelines [?, ?].

1) *Purpose specification*, requiring that the purpose for which the personal information was collected shall be stored with that information in the database.

2) *Consent*, requiring that the purpose for which the personal information was collected shall have the consent of the donor.

3) *Limited collection*, requiring that the personal information collected shall only be disclosed for purposes for which consent has been given.

4) *Limited use*, requiring that the database shall only support queries that are consistent with the specified purpose.

5) *Limited disclosure*, requiring that the personal information shall not be distributed for purposes other than those for which there is donor consent.

6) *Limited retention*, requiring that the personal information shall be retained only as long as necessary to fulfill the purpose for which it was collected.

7) *Accuracy*, requiring that the personal information stored in the database should be accurate and up-to-date.

8) *Safety*, requiring that the personal information shall be protected by security safeguards against theft and other misappropriation.

9) *Openness*, requiring that the donor shall be able to access all information about him/her stored in the database.

10) *Compliance*, requiring that the donor shall be able to verify compliance with the stated policy and the database shall be able to address any challenges.

Agrawal *et al.* sketch out a reference architecture for Hippocratic databases. An important characteristic of the architecture is that it uses some privacy metadata tables, including privacy-policies tables and privacy authorizations table. The former captures the privacy policies, while the later captures the access controls that support the privacy policies [**?**].

2.4 Anonymity

Another well-known technique to address the privacy of released data is to modify the data by removing all information that can directly link data items with individuals. Such a process is called data anonymity, which typically refers to the state of an individual's personal identity, or personally identifiable information, being publicly unknown. However, simply removing identity information, like names or social-security numbers, from the released data may not be enough to anonymize the data. There are many examples showing that even when such information is removed from the released data, the remaining data combined with other information sources may still link the information to the individuals it refers to.

To overcome this problem, approaches based on generalization and suppression techniques have been proposed, the most well known of which is based on the notion of k-anonymity, proposed by Sweeney in [**?**]. A release provides k-anonymity protection if the information for each person contained in the release cannot be distinguished from at least (k-1) individuals, whose information also appears in the release.

Two methods for achieving k-anonymity are: 1) *generalization* which replaces an individual attribute value with a broader category (e.g., Age=25 becoming $Age \in [20 - 30]$), and 2) *suppression* which replaces an individual attribute value or part of an attribute value with symbol $*$.

Table 1 shows an example of a database relation that adheres to k-anonymity, where k=2, tuples of ID=1 and ID=2 are identical, and tuples of ID=3 and ID=4 are identical.

Table 1. A 2-anonymity database relation

ID	*Gender*	*Birth*	*ZIP*
1	male	1965	0214*
2	male	1965	0214*
3	female	1964	0213*
4	female	1964	0213*

Obviously, k-anonymity can be guaranteed with the replacement of every cell with a $*$, but this renders data useless. A minimum cost k-anonymity solution suppresses the fewest number of cells necessary to guarantee k-anonymity.

2.5 Limitations of Traditional Privacy Preservation Strategies

Existing privacy preserving solutions do not fit the above main requirements.

Problems with Access Control Mechanisms Many privacy studies in database systems focus on information disclosure. However, privacy cannot exclusively be achieved by controlling accesses. Laws [**?**] and research studies [**?**, **?**, **?**, **?**] stress the importance of limiting collection and retention of information to achieve privacy. This is accentuated while dealing with smartness in context-aware systems and applications, given the lower interest of storing context information durably from a users perspective (as opposed to financial or health folders). Moreover, establishing a fair balance between information usefulness and privacy is regarded highly important [**?**, **?**, **?**]. Although limiting data disclosure and providing multi-leveled views [**?**, **?**] can meet the smartness and performance of context-aware computing, it cannot be substituted to life-cycle management.

Problems with Purposes-based Mechanisms Purpose-based techniques constitute another major attempt towards privacy protection. 1) The Platform for Privacy Preferences (P3P) [**?**, **?**] applies notice and consent practice to web browsing. Web sites describe their privacy policies (i.e., acquired data, retention period, and usage) in machine readable XML, and browsers are parameterized to accept or reject policies if hurting user's privacy settings. 2) [**?**] implements privacy principles derived from laws in ubiquitous environments using P3P and trusted devices. 3) Hippocratic database [**?**] is proposed following the principles derived from laws. It binds the responsibility for data privacy to the database, compliantly with P3P policy expression. Data is stored with the purpose for which it has been acquired and privacy parameters are derived from this purpose.

It is obvious that in many situations purposes can induce privacy parameters, defining accuracy and amount of acquired data, retention period, and access privilege. When users consent to a purpose, they consent to the associated list of privacy parameters. While this clearly shows the importance of controlling contents, such approaches hurt the smartness and usefulness properties.

First, the enforced privacy protection is good for atomic purposes (i.e., completely achieved or not at all). Privacy parameters are set to a lower bound in terms of privacy violation such that purpose can be achieved. However, while dealing with purposes which exhibit non-atomic nature (i.e., realizable partially), such purpose-based techniques are inadequate. For example, the purpose "perform smart web search" can be achieved partially, whose efficiency highly relies on accuracy and amount of granted users' context histories (e.g., current readings, accessed web sites, etc.). Note that smartness related purposes in context-aware computing are typically non-atomic, making privacy parameters difficult to figure out by service providers, and requiring rich user-based expressions such that information considered as non-sensible by the user might be granted to the system.

Second, the notice and consent practice leads to the application of the server-side privacy policy. This is even more difficult to set properly, given the usefulness property. It may lead to an important amount of non-private context information being discarded. For example, context regarding users holding weaker privacy wishes would be governed by stricter parameters defined at the server and users holding stricter policies would simply not be monitored.

Third, content life-cycle is regulated by deleting expired data after a given period or when the associated purpose is achieved. This coarse grain attempt towards data degradation is justified for data becoming completely useless after usage (e.g., credit card number), but does not suit context-aware computing environments.

Problems with Data Degradation Mechanisms Data downgrading has been further investigated to provide privacy, in balance with data usefulness, in data mining [?] and statistical databases [?]. Interesting techniques have been proposed, based on randomization [?] or data suppression and generalization [?, ?, ?, ?] to obtain k-anonym datasets. However, the downgrading process is uniform, where identical rules apply to the complete dataset. Besides, it performs only once to generate a public release, which hurts the smartness property. In addition, techniques are often calibrated for given mining algorithms or statistics computations, and require the whole original dataset [?, ?] to produce the public release, which strongly impacts the feasibility of a transposition to context-aware environments.

2.6 A Life-Cycle Policy (LCP) Model

The design of the context life cycle policy is driven by the following key considerations. To comply with the distribution property, the LCP must be self-contained, i.e., containing the whole required knowledge to be interpreted and simple to apply. To comply with the smartness property, LCP must be rich enough to avoid useless information loss (regarding privacy). In addition, the usefulness property claims for user-defined LCP. Finally, LCP should be easily transmissible to third parties to enable controlled data replication.

Context State We model context information sensed in a context-aware environment as a triplet (*ID, Time, Value*), consisting of the ID of the donor, the $Time$ when this context was acquired, and the context concrete $Value$ itself (e.g., location coordinates). Such a triplet is called a *context state*. Each element of the triplet can have different levels of accuracies based on domain generalization techniques [**?**]. For sake of simplicity, we consider that those different accuracies can be classified, given the privacy of the information they represent, as illustrated in Table 2.

Table 2. Different levels of context accuracies

Context Element	**Level-1**	**Level-2**	**Level-3**	**Level-4**	**Level-5**
Donor's ID	employee	group	department	university	
Acquisition $Time$	second	minute	hour	day	month
$Value$ (e.g., *Location*)	coordinate	room	floor	building	region

Note that although generalization domains might be a graph (non-linear), any element of the graph can be ranked regarding the privacy concern of information attached to it, depending on specific application scenarios.

The different combinations of accuracies in Table 2 thus form a cube. Each coordinate in the cube corresponds to a context state at a certain accuracy level. For example, the triplet (3, 3, 2) designates a context state, stating the donor's identifier as a department number, the time in hour, and the location as a room number.

A binary partial order relationship $\geq_a$ can be further defined in the cube to compare the accuracy levels of two context states ($s_i \geq_a s_j$), if and only if each of the three dimensions of s_i has a higher or equal accuracy level than that of s_j.

Note that the cubic representation of the data is already used, e.g., in data warehouse to represent the complex result of a given query. The main difference with the representation here is that each dimension takes different data accuracies linked to a given domain of values (more or less generalized) or to a given data type (more or less precise), ordered from the more accurate (e.g., exact coordinates for location) to the less accurate (e.g., building), instead representing an ordered set of discrete values (e.g., years) or intervals (e.g., age between 0-10) having the same accuracy.

Context State Degradation Triggered by events, the accuracy level of a context state can be consistently down degraded, forming the life cycle of a certain type of context information. According to the happening places, three kinds of events are categorized:

– *universal events*, which can be generated and thus be available in any ambient intelligent environment (e.g., a time predicate);
– *internal events*, which originate from a specific ambient intelligent environment, and are usually monitored by sensors or related to a component like a database access or a printer error;
– *external events*, which are monitored in another ambient intelligent environment, thus originating externally.

Definition 1. *The life-cycle of a context state for a certain type of context can be defined as a* deterministic finite automata *$(S, \Sigma, \delta, s_0, s_f)$, where*
- S is a set of context states in the form of a triplet $(ID, Time, Value)$;
- Σ is a set of events;
- δ is a set of transition functions $S \times \Sigma \rightarrow S$, satisfying that for a transition $\delta(s_i, e_k) = s_{i+1}$ $(s_i \geq_a s_{i+1})$;
- $s_0 \in S$ is the context starting state; and
- $s_f \in S$ is the context final state, which can be empty value ϕ, corresponding to the context state deletion. □

Example 1. Given an automata:
$S = s_0, s_1, s_2, s_3, s_4, s_f = \{(1,1,2), (1,1,3), (1,4,3), (1,3,3), (2,4,3), \phi\}$,
$\Sigma = \{e_1, e_2, e_3, e_4\} = \{$10 minutes later, leave through back door before 5pm, 8 hours later, 1 week later, 1 month later$\}$,
$\delta(s_0, e_1) = s_1, \delta(s_1, e_2) = s_2, \delta(s_1, e_3) = s_3, \delta(s_2, e_4) = s_4, \delta(s_3, e_4) = s_4, \delta(s_4, e_5) = s_f$, $s_0 = (1,1,2), s_f = \phi$.
It describes a life cycle policy example to preserve the privacy in case one decides to leave the office building earlier. Its pictorial representation is shown in Fig. **??**.

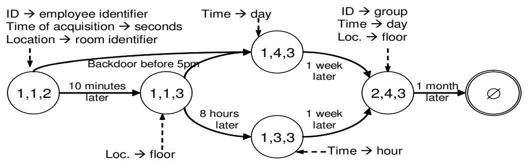

Fig. 1. An LCP automaton example

The final state of context data *Location* will store ID as the working group of the donor, $Time$ as a day, and $Value$ as the building identifier, which will be retained durably in the context database, and considered as non-private by the donor. □

As a result, the LCP-based context degradation policy can be viewed as a path in a cube using the automata model.

The One-Way Property The correctness of the LCP model in providing privacy lies in its one-way property guarantee. That is, from an already degraded context value, the system (even the DBA of the Context-DB) is not able to derive previous accurate values. However, this one-way property could potentially be violated, causing users or organizations' privacy to be violated.

In particular, compliance with this property induces constraints on the automata itself, impacts the logging process, and requires some particular techniques in the particular case of $Time$ and ID degradation.

2.7 Use Scenarios of LCPs

Organization-Oriented LCPs Consider LCPs defined by an organization (e.g., a company or a country) to preserve its own privacy, preventing useless context retention that could be subject to attacks from its competitive organizations. To achieve its privacy goal, an organization has to minimize the available (retained) data within its own information system in order to avoid potential spying. Using LCPs, an organization can parameterize its own information system to only retain context information which is strictly required in providing the services to improve its efficiency.

For example, a company could require
1) phone call redirection,
2) automatic filling-in daily timetable forms,
3) room availability forecasting for the next week,
4) statistics in terms of visibility of different teams (i.e., number of days per week a team is represented by one of its members in the organization), and so on.

To provide these services with privacy in mind, the LCP pictured in Fig. 2 could regulate employees' location information acquired by the context-aware ambient intelligent space.

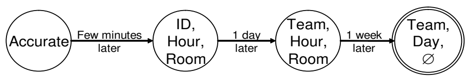

Fig. 2. An organization-oriented LCP example

Following this LCP, the context-aware system retains
1) accurate location states (employee ID, precise acquisition $Time$, and $Room$

identifier) for a few minutes (allowing phone call redirection);
2) then the exact date of acquisition is degraded to $Hour$ of acquisition (enough to allow automatic fill-in of daily timetables);
3) one day later, employee's ID is degraded to her $Team$ identifier (enabling room availability forecast for the next week), and finally,
4) one week later, identifier of *Room* is deleted (still allowing day-per-week visibility of the team at work). In this particular case, the last context state is considered as non dangerous for the organization's privacy, and can thus be retained durably in the system to enable further statistic computations and long term historical analysis.

Although this LCP is shared by all the employees of the company, it reduces the amount of context information available in the context-aware ambient intelligent space, which could be accessible in case of a spying attack conducted by a competitive.

User-Oriented LCPs Besides organization oriented LCPs, user-leveled service acceptance based on the well-known notice and consent strategy (as promoted in worldwide laws and directives) leads to defining individual LCP per user. Indeed, the list of services a given user consents to determines the LCP regulating his/her context information. To a certain extent, this LCP might serve as a quasi identifier of this user. It gives a fuzzy joining key to gather accurate and degraded context values belonging to this user when LCPs linked to data are applied.

In the spirit of notice and consent, LCPs are set by default to prevent from storing any information about the donors. For example, in a road system continuously monitoring location of cars, context states would be dropped immediately when acquired or received. Services available in the context-aware environment notify users of the context information they need for offering a given benefit. Driven by the service requirements, they ask users to consent adding intermediate context states to their current LCPs. Users, balancing the loss of privacy and the benefits offered by the service, would possibly consent to additional intermediate states. On this compromise basis, services can progressively ask for more data in exchange of additional benefits, which might lead to a rich and highly personalized LCP. Note that each user might consent to a specific pool of services.

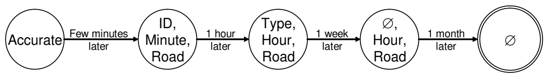

Fig. 3. A user-oriented LCP example

For example, in a car environment, some users could consent to an LCP, as shown in Fig. 3. Such an LCP leads to
1) storing the accurate context state for a few minutes to estimate car speed and

traffic overload which are necessary for advising personalized (car *ID* is stored) driving direction efficiently, e.g., avoiding traffic jams, localizing colleagues, etc.;
2) degrading the accurate location to *Road* and storing it for one hour to enable personalized forecasts (one hour history is stored), e.g., to advise user's colleagues on car pooling possibilities. Here data is retained for one hour because we assume that people susceptible to make their cars available for car pooling wait for maximum one hour to fill their cars before moving on;
3) degrading car *ID* to car *Type* and keeping it for one week to enable per type car forecasts (one week history), e.g., providing general car pooling service (e.g., estimating the best places to be given a lift) for the next week;
4) deleting the *ID* field from the context state triplet to enable general forecasts for the next month, e.g., enabling to plan road directions and avoid traffic jams in advance; and
5) finally removing the whole context state triplet.

Customized LCPs Both organization and user oriented LCPs can be refined by their owners to reflect special privacy wishes. In particular, a certain owner can complement her LCP with his/her *individual* personal requirements with extra context states and transitions.

To illustrate the shape of such an LCP, let's consider the example presented in Example 1. This LCP can be considered as containing an initial subpart either issued by an organization or in a user oriented fashion, plus a user specific subpart coping with particular preoccupations, as shown in Fig. 4. In this case, the third context state of the initial LCP subpart, referred to as (*Group, Hour, Floor*), will not be reached when the owner of the LCP leaves the building before 5pm. Context information will however be degraded to the additional user's specific state referred to as (*Group, Day, Floor*). Such a customization of the LCP would enable the owner to keep private the *hour* when s/he left her work, in case s/he left earlier than expected.

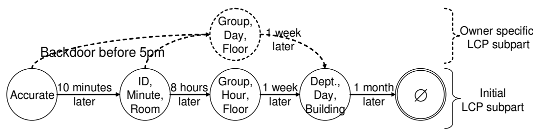

Fig. 4. A user-oriented LCP example

Two interesting remarks can be made regarding such a policy.

First, while the system knows during one week that the user left incognito by the backdoor (the tuple belongs to a different context state), the exact *hour* when she left is effectively hidden.

Second, the interest of the user to define the additional state (*Group, Day, Floor*) might be discussed compared to simply forcing to skip not enough degraded intermediate state(s). Indeed, this state offers information that does not exactly correspond to those required by services based on (*Group, Hour, Floor*). It may be useless for any application. In that case, direct degradation to the next degraded state (*Dept, Day, Building*) fulfilling the particular user's privacy requirement might be more appropriate. Here, this user defined intermediate state is considered to maximize context information accuracy in the spirit of the usefulness requirement.

2.8 LCP-based Context Privacy Protection Diagram

To reconcile privacy and smartness in context-aware computing, With the life-cycle policy (LCP) model, progressive degradation of context histories can be specified, and context-aware systems in charge only retain the desired "souvenir". Such an LCP can be viewed as a "program" bound to the context data it regulates, so that any computing component like user devices, databases and routers can "compile" it and operate the degradation.

To conform to the distribution nature of context-aware ambient intelligent spaces, as the diagram in Fig. 5 shown, a policy binder associates sensed context with corresponding user-defined LCP, regulating its life cycle. This policy binder is placed close to or even within the acquisition devices. A policy compiler situated at the context database side translates the LCP policy into a language, which is understandable at the component (e.g., SQL for a relational database). Then the component will be in charge (possibly with the help of the translator) of managing properly the context life cycle following the LCP.

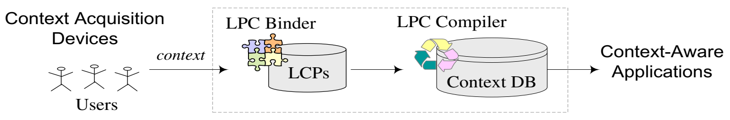

Fig. 5. Life-cycle policy (LCP) based context privacy protection

3 Performance Evaluation

to be added ...

3.1 Challenges with LCP-based Privacy Protection

While the LCP approach tackles the important privacy problem in context-aware computing, the LCP model itself also leads to a number of further research issues and open problems.

Multi-Usage The presented LCP is based on a linear generalization model for a context cube. However, since generalization/specification hierarchies are often application-dependent, the generalization trees might have several branches, potentially one required per application. The LCP model needs to be adapted to rich generalization schemas so as to cover a broad range of applications. One potential way to resolve this issue could be based on duplicating information, which must follow different generalization paths for different applications. Also other degradation techniques might be envisioned (e.g., progressively deleting bits) for specific attributes (e.g., IP addresses collected by network access points) in the context-aware soft meeting planner application.

Trustworthiness The LCP model offers a resistance to a posteriori (i.e., after LCP appliance) snooping attacks, which constitute a major threat in context-aware environments. Note that a posteriori privacy violation can be performed against individuals by malicious organisms before credit or insurance acceptance, or against companies before financial operations, insider trading, etc. For the irreversible degradation, such a protection can be achieved by only applying the LCP model once against context data.

However, it is worth mentioning that only privacy-enabled systems instead of privacy-enforced systems are addressed here. That is, without any additional security feature, trust in the underlying data management system that really degrades the data is needed. In particular, attacks can also be conducted by altering (even randomly) data involved in LCPs, replacing some LCPs with other valid (and weaker) LCPs or by a previously defined (weaker) LCP defined by the same user. How to enforce proper degradation process, e.g., by means of cryptography or secure hardware, is a very interesting and difficult open issue. Another perhaps more tractable problem would be to detect misuse at the Context-DB level, based on database audit techniques (like monitoring database events, incoming queries, and produced results). Indeed, in the above settings, nothing can prevent a malicious third party holding sufficient access privileges to continuously query the fresh (accurate) subsets of the context database in order to constitute the complete accurate history of the context-aware ambient intelligent space.

Distribution Context-aware ambient intelligent spaces incorporate many components. Moreover, several contiguous spaces might coexist. Such a distribution nature introduces many challenging issues. One of them is related to event detection (for triggering context degradation in the LCP model). Although some local events (monitored within the internal AmI space) might be broadcast to any internal recipient responsible for managing context data associated with the LCP, it is difficult to assume that external events (monitored in another context-aware space) will be. In fact, events themselves constitute parts of context information, and as such may be regulated by LCPs leading to their own progressive degradation (which can occur rapidly). This will definitely prevent intensive external broadcast. Besides, through events, some information which should not be disclosed to external ambient intelligent spaces might be revealed. In addition to event detection, distribution also incurs some other difficult issues like management of knowledge of LCP and knowledge of the appropriate generalization hierarchy for any surrounding (visitor) user, etc.

Querying Multi-Accuracy Data The LCP model leads to the management of multi-accuracy information leading to further interesting research issues, including query language and processing techniques to cope with this multi-accuracy data. Notably, a query language is necessary to enable applications to express queries involving different granularities of data provided by the context cubes, e.g., in an SQL-like language, SELECT Id [department] FROM Location_cube WHERE Value[room] = "thisRoom".

Also, accelerating computation techniques on data samples could benefit from the different accuracy levels to present to applications results with more or less precision given the current load of the database system (assuming less accurate results, based on more degraded data, are faster to compute). More generally, query processing and index techniques should be devised to tackle special properties of multi-accurate data in insertion-intensive database systems.

One-Way Data Degradation vs. Dynamic Computation While one-way (irreversible) data degradation is prompted here, it would also be interesting to investigate some other degradation schemes based on dynamic computation of current context states. For instance, one might imagine going backward along a generalization hierarchy. This could be studied by adopting a multi-accuracy access control strategy. Indeed, while traditional systems propose access control policies based on selection and projection, this approach is not flexible enough to fully satisfy donors' particular wishes when applied to pervasive environments, where control should be given at different data accuracy, or offer micro-views as shown in [**?**], and should also be able to evolve with the time. In the spirit of the LCP model, the provided accuracy may either degrade or upgrade under certain circumstances, e.g., current location can be considered as private and become accessible one week later.

Static vs. Dynamic Life-Cycle Models Addressing the limitation of static models would lead to delegate further intelligence to context-aware components (e.g., database), to enable to compute dynamically (i.e., at runtime) the most appropriate next context state given the current database content. For example, we could imagine to define degradation policies by means of k-anonymous wishes (e.g., each ten minutes, anonymize my context 5 time more, taking into account that I would prefer degrading in favor of ID, then $Time$, and then context accuracy), delegating to the system the determination of the most appropriate degraded state, given the current database content.

4 Conclusion

Context-aware computing systems continuously monitor surrounding individuals' behavior to make existing applications smarter, i.e., make decision without requiring user interaction. While this smartness ability is tightly coupled to the quality and quantity of the available (past and present) information, context linked to surrounding individuals falls under privacy directives. This chapter presented two techniques to deal with this paradox between smart context-awareness and privacy in context-aware computing environments. The first is on encrypted context information searching without decryption, so that the deployed encryption technique should on the one hand satisfy the privacy protection requirements on context information, while at the same time allow efficient manipulation of context information without loss of confidentiality. The second is on life-cycle management of privacy-sensitive context information, where life-cycle policies are bound to context data to regulate the progressive degradation of the accuracy of context information.

References