# Generating Privacy Constraints for Assistive Environments

Hannes Grunert
Database Research Group
University of Rostock
18051 Rostock, Germany
hg(at)uni-rostock.de

Andreas Heuer
Database Research Group
University of Rostock
18051 Rostock, Germany
ah(at)uni-rostock.de

## ABSTRACT

Smart environments produce large amounts of data by a plurality of sensors, which constantly track our activities and desires. To support our daily life, assistive environments process these data to calculate our intentions and future actions. In many cases, more information than required are generated and processed by the assistive system. Thereby, the system can learn more about the user than intended. By this, the users' right to informational self-determination is injured, because they lose control how their data is used.

In this paper, we present a model to let the user formulate requirements to protect his privacy in smart environments. These requirements are transformed into multiple integrity constraints, which ensure privacy.

## Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Query Processing*; K.4.1 [**Computer and Society**]: Public Policy Issues—*Privacy*

## General Terms

Privacy Enhancing Technologies, Database Systems

## 1. INTRODUCTION

Assistive systems are designed to support the user at work (Ambient Assisted Working) and at home (Ambient Assisted Living). Sensors collect information about the current situation and actions of the users. These data are stored by the system and linked to other data, for example social network profiles. Based on the obtained information, preferences, patterns of behavior and future events can be calculated. Furthermore, intentions and future actions of the users are derived, so that the smart environment can react independently to satisfy their needs.

Assistive systems [11] often collect much more information than needed. In addition, the user usually has no or only a very small effect on the storage and processing of his personal data. In extending an assistive system by a data protection component, which checks the privacy claims of the user against the required information of the system, this problem can be resolved.

Two main principles of data protection are data minimization and data avoidance. This includes the design of information systems as well as the data processing itself. By means of a data-avoiding sharing of sensor and context information towards the analysis tools of the assistive system, not only the privacy-friendliness of the system is improved. The privacy claims and the required information of the analysis tools can be implemented as integrity constraints in the database system that stores the data collected by the sensors. Due to the integrity constraints, the necessary algorithms for preprocessing and anonymization can be run directly on the database.

The privacy module examines the users' requirements and compares them with the required information of the system. Instructions for selection, compression and aggregation are generated so that only the required data will be stored and processed in the system. A query can be decomposed into multiple subqueries which can achieve sub-goals already at the sensor level. Thus, a transfer of all data to a superior computing unit, which executes the actual analysis algorithms, can be dispensed.

The aim of our work is to develop a privacy-friendly query processor that implements the aspects of data minimization and data avoidance. The processor is integrated within the PArADISE [1] framework. The evaluation of our framework is based on the sensor and context information collected at the Smart Appliance Lab of the graduate program MuSAMA [2].

In this article we present the language used to formulate the data protection requirements. The rest of the paper is organized as follows: In the second section we present the Privacy Policy for Smart Environments (PP4SE) and mechanisms for its automatic generation. Section three gives a brief overview of the related work. Finally, we draw our conclusions and an outlook on future work.

[1] Privacy-aware assistive distributed information system environment
[2] Multimodal Smart Appliance ensembles for Mobile Application


*PETRA* '15, July 01 - 03 2015, Island of Corfu, Greece.
ACM 978-1-4503-3452-5/15/07...$15.00.
DOI: http://dx.doi.org/10.1145/2769493.2769542

# 2. POLICIES

In order to collect, to process or to use personal data, the system has to specify (according to [1]):

1. the purpose and duration of the contract,
2. the extent, nature and purpose of the proposed survey,
3. the processing, use and type of data,
4. the range of stakeholders and
5. the origin of the data (provenance)

One aspect of the PArADISE framework intends to support the user of assistive systems to maintain his privacy. Therefore, the privacy-claims of the user have to be defined in a policy. In this chapter, we introduce the two main types of privacy constraints and their representation as a privacy policy for smart environments (PP4SE).

## 2.1 Sensitive Data

Privacy can be divided into two categories: anonymity and hiding of specific information.

### *2.1.1 Stay anonymous*

For the protection of personal data, there exist concepts such as k-anonymity [8] and l-diversity [4].These concepts divide the attributes of a relation into keys, quasi-identifiers, sensitive data and insensitive data. The aim is that the sensitive data cannot be clearly assigned to a particular person. Tuples can be uniquely determined by key attributes, so they should not be published together with sensitive attributes under any circumstances.

PArADISE implements concepts for relational databases to anonymize data as soon as possible. Similar to antivirus programs, the framework integrates new models as soon as possible to ensure privacy when new variants of attacks arise. Ubiquitous environments often use resource-limited devices with low CPU frequency or main memory. To cope this problem, we try to optimize the privacy-preserving algorithm with regard to such devices.

One example for this is our approach to find quasi-identifiers in high dimensional datasets. Quasi-identifiers are used to define which sets of attributes allow the re-identification of persons or activities, even if key attributes are removed. The identification of a tuple in a relation using these attribute combinations is still possible, but not 100% sure. QIs can be part of a key, but also combinations of sensitive and insensitive data.

Due to the high dimensionality and the large amount of data generated in an assistive system, it is not easy for an inexperienced user to recognize which attributes compromise his privacy. The same applies for the derivation of additional information from the existing data. In [2] we introduced an algorithm which reduced the time to find all quasi-identifiers by more than 95% (in contrast to the algorithm proposed by [6]). By this, the workload of sensors and other resource-limited devices is not affected too much. The method alternates between a bottom-up- and a top-down-approach and shares the knowledge about (negated) quasi-identifiers. Per calculation step either the top-down (testing attribute-combination with many attributes) or bottom-up (small combinations) method is executed and the results of the step are passed to the other method.

Quasi-identifiers and key attributes allow the unambiguous identification of an object, such as a person or an action. These attributes make it possible to combine information from different tables, databases or with background knowledge. The key property of an attribute is usually defined within the database design process. In most cases, system-generated artificial attributes are created for keys, but several existing attributes can be recognized as a key. Keys should not be passed; they can be linked to information outside from the system and by this, new information on the affected object can be determined.

Sensitive data is data that needs to be protected. It depends on the application domain and the affected person to whom the data relate, which data is classified as sensitive. In contrast to sensitive data, insensitive data are those which do not necessarily have to be protected.

### *2.1.2 Protection of secrets*

In some assistive systems, it is necessary that the user has to be identified; so the data is not anonymized. Nevertheless, the user may not want to reveal all information about him or her. In this case, the user has to define his privacy requirements into a policy.

For an inexperienced user, it is not easy to make meaningful privacy settings. To make matters worse, in ad-hoc environments, the sensors and processing computer and the algorithms used can be replaced without the user even noticing it. For these reasons, it is important that the privacy component of the assistive system can generate useful integrity constraints for the user.

To overcome this problem, we are generating privacy policies for new devices based on the settings the user has specified for older devices before. We realize this in three steps. In the first step, schema mappings [5] between the new device and all known old devices are generated. The privacy settings of the best fitting devices are adopted for each mapped attribute. For those attributes on the new device without a compatible attribute, data mining approaches (clustering, classification) are used to find out which attributes with similar privacy settings are used together. From these attribute groups we adopt the settings for the unmapped attributes. In the third step, all attributes without a privacy mapping are set to *PRIVATE*, a state where the attribute value cannot be accessed.

The generated privacy settings for the new application/sensor are only of preliminary nature. It is up to the user to accept these settings or to modify them later to meet his desires.

## 2.2 Privacy policies for smart environments

Hitherto, security models and privacy policies focus on how whole data records can be protected from unauthorized ac-

cess. Adding a more fine-granular access control allows the user to specify which information is revealed and how the data are processed. Data records can be modified by selection, projection, compression, aggregation and feature extraction (e.g. on motion images) to reduce the amount of personal information send towards the tools for evaluation and analysis.

Our privacy policy is based on the draft of the W3C for Privacy Preferences Platform [10], but leaves out browser-specific details, such as the management of cookies. In return, our policy provides additional information for configuring data streams, such as the allowed query interval and possible aggregation levels.

### 2.2.1 Internal Representation

In this chapter the policy is explained based on a small example, which is illustrated in Figure 1. The formal definition is given in the next section.

```
<policy>
 <apps>
  <application id="MyFridge">
   <appDescription>
    This is a smart fridge.
   </appDescription>
   <modules>
    <module id="CheckContent" type="basic"
       required="yes">
     <moduleDescription>
      This module will check which
        amount of food is remaining.
     </moduleDescription>
     <attribute-list>
      <attribute name="amount">
       <allow>true</allow>
       <third-party-access>false</third-party-access>
       <condition>FAT <= 40</condition>
      </attribute>
     </attribute-list>
    </module>
    ...
   <modules>
  </application>
 <apps>
</policy>
```

**Figure 1: Example for a privacy policy**

For internal representation and data exchange, we store the policy as an XML file. A policy is a set of applications (*apps*). An application can be a stand-alone computer program, a background process of an operating system or even a smart environment as a whole. Each application consists of a unique ID (in the example: *MyFridge*), which allows unique identification by its name. In addition, an application has a description and a list of functions (*modules*).

A module of an application is a well-defined task to solve a specific problem of the entire system. For example, a fridge has the basic functionality to check its contents (in the example: *CheckContent*) to determine how much food remains. Another module may implement the automation of buying new food. A module is characterized by its ID and its type. Each module consists of a description and an indication if the functionality is required to execute the application.

All modules provide an *attribute-list* containing all needed attributes. Attribute-lists can be connected with boolean operators to provide different alternatives if a user does not want to reveal a specific combination of personal information. An attribute is identified by its *name* and persists of different privacy settings. The access conditions for the application are controlled by the *allow*- (the application itself) and *third-party-access*-tag (to shrink continued processing). The user can specify several *condition*s under which the information is disclosed. Additionally, settings regarding *aggregation* and *confidentiality* (privacy-level of detail) are specified. In the example, the application has access to the amount of food remaining for every item in the fridge, except of the food which has more than 40% of fat.

The separation of functions and applications is important, because this ensures that information can only be used for the intended purpose. Thus, the user may share his data for personalized advertisements. On the other side, the user does not want this information to be transmitted to his insurance company, so that his contribution rate will be adjusted according to his eating habits.

Due to space limitations, we cannot present the whole policy in detail. For more details, please refer to [3]. The policies are constructed by modifying the underlying DOM-structure of the XML-documents. The user manages the policies through a simple graphical interface.

### 2.2.2 Formal definition

Our privacy policy model is a 9-tupel $P := (APP, M, ATTR, C, I, A, AGG, ANO, f)$, where

- APP := a set of applications
- M := a set of modules
- ATTR := a set of attributes
- C := a set of conditions
- I := a set of query-intervals
- A := a set of access conditions
- AGG := a set of aggregation functions
- ANO := a set of anonymization functions
- f := a partially defined function, which maps a combination of an application $app \in$ APP, a module $m \in$ M and an attribute $attr \in$ ATTR to a set of conditions $C' \subseteq$ C, as well as a query interval $i \in$ I, several access conditions $A' \subset$ A, an aggregate function $agg \in$ AGG and an anonymization function $ano \in$ ANO.
  $f := (app, m, attr) \rightarrow (C', i, A', agg, ano)$

The terms application, module, attribute and (access) condition have been introduced above. The term interval is related to data streams and their continuous collect of information. By specifying an interval, the user can decide in which frequency his data can be accessed. Aggregation functions allow the user to hide raw data. Typical functions are sums, minimum and maximum values and medians as well as complex, statistical constructs like correlation coefficients and regression lines. By selecting an anonymization

function f, the user can decide which privacy model is used in the postprocessing stage. If no method is selected, an appropriate one is selected automatically.

Having selected the function f, the privacy requirements are transformed into Multiple Access-control Generated Integrity Constraints (MAGIC) for specific query languages and concrete (database) systems.

# 3. RELATED WORK

Usually, simple access control lists (ACL) are often used for the formulation and enforcement of privacy rights. ACLs define the access rights only for files, but not for content. Besides pure rights management, privacy policies can be used as an advanced concept that allows a more detailed definition of privacy claims.

The World Wide Web Consortium (W3C) is proposing two standards for the formulation of privacy policies. On the one hand, there is the Platform for Privacy Preferences Project [10], a markup language that is intended for use in browsers. The user can generally define what kind of websites can be accessed. The website provider can formulate what data he stores, e.g. by means of cookies, about the user. If these requirements contain contradictions, the user is warned.

In addition to the P3P, the W3C has suggested the Enterprise Privacy Authorization Language [9], another language that is specifically designed for the exchange of data in business environments. Both language proposals are based on the XML format and provide a rule-based access mechanism in which roles and conditions are formulated to grant or deny write and read access rights.

Besides the above mentioned W3C standards, there exist many other language proposals, such as the eXtensible Access Control Markup Language [7]. These do not differ substantially from the existing language proposals.

# 4. CONCLUSIONS

The protection of privacy in ubiquitous environments is a challenging task. A variety of sensors capture every moment of our life. In this paper, we present a method to formulate privacy claims on an assistive system, so that we become the master of our own personal data. By reducing and preprocessing the data generated by the sensors and processed by the system, privacy can be ensured in smart environments. Powered by the data protection profiles of the users, the acquisitiveness of assistive systems is limited. Our privacy-aware query processor gives a comprehensive tool to ensure privacy by analyzing queries and optionally rewrites them.

At the present time, we integrate existing data protection techniques and analysis functions (in the context of student projects) into the query processor. It is recommended that further research should be undertaken in the following areas:

- Automatic comparison between the privacy requirements of the user and the required information of the system.
- Anti-virus programs offer protection against the newest malicious software. Can this procedure be adapted to data mining technologies and privacy protection mechanism?

Further studies, which take the comparison of the privacy constraints and the analysis functions into account, will need to be performed. We are in the process of investigating the transformation process of complex analysis functions into SQL-queries.

Unfortunately, many aspects and details could not be included in this article due to reasons of space. A longer version of this paper is available as a technical report [3].

# 5. ACKNOWLEDGEMENTS

Hannes Grunert is funded by the German Research Foundation (DFG), Graduate School 1424 (Multimodal Smart Appliance Ensembles for Mobile Applications - MuSAMA). The authors gratefully acknowledge the constructive comments of the anonymous referees.

# 6. REFERENCES

[1] Federal Republic of Germany. German Federal Data Protection Act, 2010. in german.

[2] H. Grunert and A. Heuer. Big Data und der Fluch der Dimensionalität: Die effiziente Suche nach Quasi-Identifikatoren in hochdimensionalen Daten. In *Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken)*. `http://ceur-ws.org`, 2014. in german.

[3] H. Grunert and A. Heuer. Generating Privacy Constraints for Assistive Environments. Technical Report ISSN 0944-5900, CS-01-15, Institut für Informatik, Universität Rostock, May 2015.

[4] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[5] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 117–128. IEEE, 2002.

[6] R. Motwani and Y. Xu. Efficient algorithms for masking and finding quasi-identifiers. In *Proceedings of the Conference on Very Large Data Bases (VLDB)*, pages 83–93, 2007.

[7] OASIS. eXtensible Access Control Markup Language (XACML) Version 3.0. `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`, 2013.

[8] P. Samarati. Protecting respondents identities in microdata release. *Knowledge and Data Engineering, IEEE Transactions on*, 13(6):1010–1027, 2001.

[9] W3C. Enterprise Privacy Authorization Language. `http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/`, 2003.

[10] W3C. Platform for Privacy Preferences (P3P) Project. `http://www.w3.org/P3P/`, 2007.

[11] M. Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.