Supporting Patients’ Privacy Preferences Using Aspects

Kung Chen*¹ Da-Wei Wang*²
*¹National Chengchi University, Taiwan, *²Academia Sinica, Taiwan
Correspondence: *¹chenk@cs.nccu.edu.tw, *²wdw@iis.sinica.edu.tw

With the spread of electronic health records, patients’ privacy concerns rise greatly. In response, many organizations advocate that healthcare information systems (HIS) should have an opt-in capability and a way for people to segment sensitive information, if they choose to. Currently, employing a proper access control mechanism to protect patients’ electronic health records is a well-accepted discipline in HIS development. However, the design of such mechanisms hardly includes the requirement of supporting patients’ preferences regarding the use of their personal information. It is therefore highly desirable to extend a HIS’s access control to handle patients’ privacy preferences. On the other hand, as the principle and practice of patients’ privacy preferences are still emerging, instead of replacing existing mechanisms with new ones, techniques for adapting existing mechanisms to quickly reflect patients’ privacy preferences can be a worthwhile solution. In particular, we argue that aspect-oriented programming (AOP) can be part of the solutions and has the potential to provide fine-grained privacy protection to cater to the privacy needs of each individual. Aspect-oriented approach enables separation of concerns which are better designed independently, but must operate together. We propose an aspect-based preference management framework that collects and manages patients’ preferences independently yet can integrate with the underlying HIS to support patients’ privacy preferences effectively. The proposed mechanisms are loosely coupled with the underlying system. It is thus easy to adapt them and employ them for migrating existing systems to support patients’ privacy preferences.

Key words: Privacy, Access control, Patient preferences, Aspect-oriented programming

1. Introduction

Healthcare information is valuable yet sensitive. Electronic health records can make health care safer and more efficient, but also threaten personal privacy more. Exchange of electronic healthcare information can harness even greater benefit yet pose greater threats to individual privacy. Although it is always considered that information technology is one of the major causes of our losing privacy, information technology can also be an enabler of our privacy. Indeed, one important principle regarding utilizing personal identifiable health information is consent or choice, which means that each individual should decide when and how her individual information can be used by what purpose. Obviously, it would incur tremendous administrative cost, should this principle is really implemented to pertain to individual options in a paper based system. In contrast, information technology can implement this principle with acceptable overhead.

On the other hand, such advancement in priva-
cy rights can hardly be achieved in one step. Most people do not have a clearer picture about how ones’ choice will be honored. It is conceivable that even the whole social framework as well as the healthcare information systems (HIS) have to go through a transition period before most details settled. Unfortunately, healthcare information systems have to be very reliable and constantly patching or redesign the system is rarely an option. How to go through the transition period so that the stability of information systems demanded by healthcare organizations and the gradually progression of social framework can co-evolve is a major problem.

In this paper, we propose to apply aspect-oriented programming (AOP) to implement privacy enforcement and auditing mechanisms, so that individual choices can be well respected without reengineering the whole health information system. Aspect-oriented approach enables separation of concerns which are better designed independently, but must operate together. As the patient-consent principle cannot be realized by traditional access control available in most healthcare information systems, we argue that AOP is a good approach to enhancing access control with privacy concerns. In particular, we present an aspect-based privacy management framework that collects and manages patients’ preferences independently yet can integrate with the underlying HIS to support patients’ privacy preferences effectively. As a result, the impacts on the underlying HIS could be reduced significantly.

The remainder of the paper is organized as follows. Section 2 briefly reviews the basics of AOP and the relevant features of AspectJ, our implementation language. Section 3 presents the design of the privacy framework for enforcing and auditing patients’ preferences. Section 4 sketches the implementation design of the privacy framework, focusing on the key design considerations. Section 5 describes related work. Section 6 discusses the proposed framework and sketches future work.

2. Preliminaries: AOP and AspectJ

Aspect-oriented programming aims at modularizing crosscutting concerns such as profiling and security that are generally spread throughout the components of a software system and tangled with core functionalities. In AOP, a program consists of many functional modules (base program) and some aspects that encapsulate the crosscutting concerns. An aspect module provides three kinds of specifications to realize crosscutting concerns of the base program. The first kind defines pointcuts, which select a set of well-defined points in the execution of a program, called join points, designating where to crosscut other modules. The second one defines advice, which is a piece of code associated with a pointcut that will be executed when any join point in the pointcut is reached. The third kind introduces inter-type declarations, which define crosscutting concerns that operate over the static structure of type hierarchies in the base program.

Basically, advice specifies changes to the behaviors of functional modules they crosscut whereas inter-type declarations enhance the static structure of types referenced in the functional modules. The complete program behavior is derived by some novel ways of composing functional modules and aspects according to the specifications given in the aspects. This is called weaving in AOP. Weaving results in the behavior of those functional modules impacted by aspects being modified accordingly.

AspectJ is a seamless aspect-oriented extension to the Java programming language. It provides a pointcut language through a powerful join point model. Typical join points in AspectJ are method and constructor execution, and field access. The context at a join point, called this Joint
Point, is the environment information that can be exposed and passed to an advice via a pointcut. Advice in AspectJ is an anonymous method bound to a pointcut and tagged with one of the three keywords: before, after, or around. Before advice and after advice are executed before and after the method(s) selected by their pointcuts, respectively. The case for around advice is more subtle; it will be executed while the selected methods are suspended. Inside the around advice, we may choose to resume the intercepted methods by calling the special built-in method proceed, or simply bypass their execution. In AspectJ, the weaver tool is tightly integrated into the AspectJ compiler, ajc, and performs the weaving along with the compilation of the aspects into bytecode.

Besides, the inter-type declarations in AspectJ allow programmers to declare new methods and fields that are necessary to implement some new capability, and associates the methods and fields to the existing classes. Thus one can enhance an existing base program to a great degree without modifying it directly. Furthermore, AspectJ also allows aspect inheritance, abstract aspects, and abstract pointcuts. We can write an aspect with abstract pointcuts and abstract methods. A sub-aspect then extends the abstract aspect and defines the concrete pointcuts and methods. Such abstract aspects open the way to build generic aspects that are essential to an aspect framework.

Our proposed scheme takes full advantages of the above features of AspectJ. The one class and three aspects in Figure 1 outline how we our approach works. Here the class, PData, contains patient data that must be protected. The aspect, PrivacyGeneric, illustrates the generic structure of a privacy enforcing aspect. It is an abstract aspect because the method, getPatientPref, is abstract. Its around advice (Line 07~13) monitors the execution of the access control module via the pointcut PDP. When an access request is granted, the advice will get the intended purpose of use and the purposes consented by the underlying patient on the contents of PData. If they match, then the execution is resumed; otherwise, the access request will be denied. Besides, there are two sub-aspects of PrivacyGeneric, namely Privacy1 and Privacy2. The first sub-aspect simply defines the missing method, but the second sub-aspect is more complex. It uses inter-type declaration

```java
01 class PData { ... }//data to protect
02 abstract aspect PrivacyGeneric {
03     abstract Purposes getPatientPref(Context co); //to be defined
04     Purposes intended; // purpose of use
05     // a named pointcut targeting at the policy-decision point of access control
06     pointcut PDP(Context co) : execution boolean AccessControl.PDP(Context xo);
07     void around(Context co) { PDP(co) { // //advice monitoring PDP
08             boolean accessCheck = proceed(); // resuming access control check
09             if (accessCheck) { //access granted
10                 intended = ActionMgr.getActionID(); //via action ID
11                 if (match(getPatientPref(co), intended)) return true;
12                 else return false; AuditMgr.log(co);
13             } // end around
14         }
15         aspect Privacy1 extends PrivacyGeneric {
16             Purposes getPatientPref(Context co) { return PreferenceMgr.getPatientConsented(co.getPID(), PData.class);
17         }
18         }
19         aspect Privacy2 extends PrivacyGeneric {
20             private Purposes ps, PData.newP;//add a new field
21         after (PData po): execution (PData.newP[]) & this (po) { ps = po.newP = PreferenceMgr.getPatientConsented(po.getPID(), PData.class); }
22         Purposes getPatientPref(Context co) { return ps; }
23         }
24     }
25 }
```

**Figure 1** An Example of Aspects in AspectJ
mechanism to add a new field to the class \texttt{PData} (Line 20), and employs after advice (Line 21–22) to fill in that field after an instance of \texttt{PData} was created.

3. The Privacy Framework

Currently, employing a proper access control mechanism to protect patients’ electronic health records is a well-accepted discipline in HIS development\(^6,7\). However, the design of such mechanisms hardly takes patient privacy requirements into account. Indeed, privacy protection goes beyond the traditional view of access control by requiring some additional complexity. There are complex questions of who controls the data about us and how they are used. In making access control privacy aware, the recent trend is that permission to use any sensitive data should be linked to the purpose of use\(^8\). For example, a doctor may have permission to hold personal data for operational health care delivery, but not for other purposes such as research, or health product promotion. Moreover, patient consent should be checked, and each data subject may make different decisions regarding which parts of his/her personally identifiable information (PII) can be accessed by who under what intent. In other words, the decision on disclosing a patient’s PII is extremely fine-grained: it does not only depend on the data contents and purpose of use, but also is patient-specific.

On the other hand, we must also consider the proper balance between patient safety and patient privacy in exceptional situations. In particular, during an emergency, an attending physician needs to bypass routine access control restrictions to provide timely treatment without any delay due to administrative or technical complexities. This is often referred to as the \textit{break glass policy}\(^9\). Yet, once such privileged requests were allowed, the access control system should create an audit trail, executed as an obligation, for administrative review. However, auditing and reporting is difficult to manage. Each application in a HIS usually creates logs in an application specific way and may not provide audit trails detailed enough for privacy auditing purpose. Preparing an audit report from such logs is not only time-consuming but also imprecise. Thus, it is desirable to have a centralized auditing mechanism that can create required logs for all requests and accesses to PII data.

The goal of our framework is to enhance access control to patients’ health records with enforcement and auditing of patients’ privacy preferences regarding the use of their personal information. The main difficulties in achieving this lies in the fact that, in current practices, privacy polices and patient consent are usually added after the HIS has been built. Hence, even though it is possible to collect patients’ preferences for controlling access to their health records afterwards, it is rather difficult to enhance the working access control system yet being able to accomplish it with little impact on the structure of the underlying HIS.

Based on our prior experience\(^10,11\), we claim that we can further leverage the aspect technology to develop a privacy framework to address the issues described above. The proposed privacy framework consists of three main components, namely action purpose manager, privacy aspect and patient preference manager. Figure 2 displays the structure of the framework, and highlights the interaction between the framework and the underlying access control system. The remainder of this section describes how they work together to provide privacy-aware access control and the next section will sketch how to implement this privacy framework.

For the purpose of this work, we follow the industry practices such as XACML\(^12\), and assume the existence of the policy enforcement point (PEP) and policy decision point (PDP) in an access control system. All user requests to access a
patient’s data will be intercepted by the PEP and passed to the PDP for permission check. The proposed privacy aspect interacts with the access control system via the PDP, namely choosing the PDP as the join point for advice weaving. The core task of the aspect is to monitor the result of PDP and perform the enforcement and auditing of patient preferences if necessary. Specifically, if the PDP grants an access request, the privacy aspect will then take charge to ensure that the intended use of the data matches those consented by the patient. If they do not match, the privacy aspect can override the decision made by the PDP, which in turn will direct the PEP to reject the access request. Moreover, regardless of the matching result, the privacy aspect will log any access activity in detail.

To operate effectively, the privacy aspect must acquire the purpose of use underlying an access request and the purposes consented by a patient for accessing the data under request. Since these are both new to the access control system, we employ two other components to handle them. One is the action purpose manager responsible for assigning a purpose of use to any access action that operates on a patient’s health record. This assignment should be carefully prepared by some security and privacy task force and entered into the manager by a privileged system administrator. Afterwards, when an access request is submitted, the privacy aspect will demand the action purpose manager to return the intended purpose for the requesting action.

The other is the patient preference manager, which is also a centralized module that collects and manages patients’ privacy preferences. Here the privacy preferences are specified in terms of consented purposes of use on PII. In particular, a patient (data subject) can dictate what purposes of use are allowed for each kind of PII in the health record. Later, when a data subject’s PII are requested for access by a data requester, the privacy aspect will invoke the preference manager to retrieve the data subject’s consented purposes for the requested PII and associate them with the PII data.

As can be seen easily from above description, only the privacy aspect needs to have direct interface with the underlying access control system. Moreover, the interface is achieved without the involvement from the system since the privacy aspect will be integrated with the underlying PDP via an aspect weaver. In other words, the behavioral adaptation of the original access control sys-

![Figure 2 The Privacy Framework](image-url)
Obligations of data collectors are also considered an essential part of privacy policies. We shall discuss them in the last section.

4. Implementation Design

This section outlines our implementation scheme of the proposed framework, focusing on a few key design aspects that are essential to the framework.

4.1 Privacy Policy and Structural Representation of Patient Privacy Preferences

We start with the structure and form of privacy rules we aim to enforce and audit, and then derive the structural representation of patients’ privacy preferences. As far as we know, there are no commonly accepted standards in the healthcare domain in this respect. Yet many projects have been launched towards the goal of establishing an interoperable format for specifying privacy rules. For example, the RBAC Security and Privacy Vocabulary Project of HL7 is such an endeavor. Among other things, this project focuses on developing a standardized Privacy and Consent catalog for specifying patient preferences in terms of purposes of use. Outside healthcare domain, there are several emerging standards for specifying privacy rules. For example, we have P3P, EPAL, and XACML 2.0 privacy policy profile, just to name a few. They all take the form of privacy-enhanced access control rules and emphasize the intended purposes of use. For the purpose of this work, we follow EPAL and assume the following structure of privacy rules: “allow or deny actions on data categories by user categories for certain purposes consented by the data subject under certain conditions.” As an example, consider the following privacy rule adapted from:

Allow read action on a patient’s blood lab results by blood disorder researchers for the purpose of medical research consented by the patient if the blood disorder researcher is not from the same zip code of the patient.

Such a rule format is not only fine-grained but also privacy-aware. Thus we think it suits our purpose well.

After choosing the format of privacy rules, we proceed to the structure of specification for patients’ privacy preferences. Ideally, we could simply use the privacy rules for patients to specify their preferences, leaving full control to the patients. However, this is not feasible in reality for practical reasons. Basically, the structure of these rules is very fine-grained and complex, and hence patients will find it difficult to make decisions without more information or assistance. Moreover, physicians may not accept the idea of leaving full control to patients. In particular, the conditions item is simply too broad to specify by the patient alone. Instead, the item is more appropriate for internal access control purpose and thus be specified by the healthcare organization or legal authorities.

The other major issues concern the granularity of the items for patients to control, namely action types, data categories, user categories and purposes of use. Eventually, these items should all follow some standardized vocabularies of appropriate granularity. For now, we argue that we should consider the granularity issues more from a practical point of view by starting from coarse granularity and moving towards fine granularity gradually as we accumulate more experience in privacy protection matters. Here, the P3P privacy standard for Web users may serve us as a refer-

---

1Obligations of data collectors are also considered an essential part of privacy policies. We shall discuss them in the last section.
Table 1 Structural Representation of Patient Privacy Preferences

<table>
<thead>
<tr>
<th>data_category 1</th>
<th>data_category 2</th>
<th>data_category 3</th>
<th>data_category 4</th>
<th>data_category 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_category 1</td>
<td>purpose 1, purpose 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>user_category 2</td>
<td>purpose 1</td>
<td>purpose 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>user_category 3</td>
<td>purpose 2, purpose 3</td>
<td>purpose 2</td>
<td></td>
<td>purpose 1</td>
</tr>
</tbody>
</table>

ence point for coarser granularity. In P3P, privacy rules are of the form: “Allow Users to Use PII types for Purposes if Conditions if Consent.” We note a few granularity limitations in P3P. First, there is no clear indication of user categories. Second, the only action is use. Third, the PII types and purposes are pre-defined and not designed for the healthcare domain.

For the purpose of this work, we suggest that the patient preferences be specified in a form that is a little more fine-grained than the P3P standard. In particular, there are two items to consider for extension. The first item is the user category. We propose to allow a simple category of users that may access a patient’s PII. But, note that the user category should not be some fine-grained categories often used in a role-based access control (RBAC) system. In general, such user categories do not mean much for ordinary patients. To them, any practitioners from the healthcare organization treating them are simply healthcare practitioners, regardless of their job titles or functional roles. The user categories such as roles are more appropriate for instituting rules for internal access control purpose. Instead, a category covering broader kinds of users, internal or external, such as primary care team, hospital staff, government agencies, and insurance companies, will be easier for patients to comprehend. The second item is the action type. In some cases, the general term of “use” may be too vague to specify the constraint. On the other hand, detailed action types may also confuse the patients. Hence, when necessary, we suggest using high-level types of actions such as read, modify, create, and append as an enhancement.

As to data category and purposes of use, they should be simple for patients to understand and feasible for system developers to implement. Specifically, if the contents of a data category follow or map directly to the structure of a patient’s healthcare data, it will be easy to use metadata tags to associate patients’ data with the category items. Some typical purposes of use include advice, care, research, governance, and product promotion. In summary, we propose to use a matrix structure to capture patients’ privacy preferences as shown in Table 1.

4.2 The Privacy Aspect

Given the structural representation of patients’ privacy preferences, we now consider the implementation of the proposed privacy aspects for enforcing those preferences. The main concern is to ensure that the privacy aspects have access to all necessary privacy context information of the data request so that the patient’s privacy preferences can be enforced and audited. Furthermore, although we focus on the purposes of use, we also anticipate extra conditions that need to be checked as required by the format of above privacy rules. Therefore, the context information required is not limited to the intended purpose of the action and the purposes consented by the patient; other items such as data requester identity, action identity, data category and data subject identity are also essential. We choose the PDP of the underlying access control system as the join point to weave in the privacy aspect. The reason should be fairly obvious. In order to make the decision on an
access request, the PDP must have access to all required access control context information, which for the most parts is in common with the privacy context information. Hence, as the privacy aspect targets at the PDP, such context information can be easily accessed in the aspect via the join point API as described in Section 2.

As to the purpose information, the intended purpose of an access action is easy to acquire. The action purpose manager is a global object which keeps a hash table that maps all actions to their intended purpose of use. Thus the privacy aspect can simply send the access action’s identity to the action purpose manager and get the intended purpose in return.

On the other hand, the case for the consented purposes is more involved. We have to distinguish between two kinds of access requests: single record access and multiple records access. For single record access, as the access action will pass the patient’s identity information along with the request, the privacy aspect will have no difficulty in getting the patient’s consented purposes. It simply sends the patient’s identity and other information such as data category and requester identity to the patient preference manager, and the purposes consented for this kind of access request will be returned. However, if the request is a general query that selects multiple patient records satisfying certain conditions, the situation is more complex and deserves more discussion.

4.3 Accessing Multiple Health Records

Retrieving multiple electronic health records simultaneously is essential to the secondary use of health data, such as research. The central issue of secondary use of healthcare data is identifiability. Basically, if a dataset is de-identified, i.e., individual’s identity can not be identified with the dataset, the dataset can be utilized. However, the concept of identifiability itself is a fuzzy one, and there is a degree of identifiability\textsuperscript{19,20}.

For our purpose, let us assume that identifiability is a function of the dataset and given a dataset its identifiability is a value between zero and one. We also assume that a person’s preference to be included in a dataset for research purpose depends solely on the identifiability of that dataset.

It is conceivable that the goal of the research and some other factors can also affect one’s preference and they can be incorporated if necessary. Immediately, we can see that there is a potential dilemma: we cannot compute the grade of identifiability of the dataset without gathering the dataset first, yet, to compute the identifiability of the dataset, we “should” need the data. Let us assume that people all agree to contribute their data for the computation of identifiability of a dataset and come back to the issue later.

With above setting, we can see that we can not decide if one’s privacy preference is respected before gathering the data for computing the identifiability. Casting in our privacy framework, this issue becomes that, when retrieving multiple health records via a general query, the privacy aspect will not have the identities of the qualified patients to get their consented purposes. To resolve the issue, we provide another scheme for getting the patients’ privacy preferences. Basically, we take a post-processing approach: conducting the privacy enforcement task after the requested patient records have been fetched but before returning them to the user. It can then works by filtering out the records whose owners do not consent the access request.

The main difficulty is how to associate a patient’s PII with the purposes of use consented by the patient. Our scheme consists of two parts. The first part adds a purpose field to any class which includes a data subject’s PII to protect. This is achieved through the inter-type declaration mechanism of AspectJ. The second part fills in the purpose field so that the link of a patient’s PII with its consented purposes of use is estab-
lished. This is conducted by an object construction advice which is triggered right after any object is instantiated from those classes with PII. Figure 3 displays the pseudo AspectJ code snippet we employ to accomplish the two parts.

Specifically, the aspect GetMultipleRecords employs a post-processing around advice (Line 3–13) to scan and filter out unwanted records from the retrieved dataset. As it temporarily allow retrieving the records that satisfy the query condition, a collection of objects that embody the healthcare records will be created by the underlying HIS system. Hence, another piece of advice (Line 18–20) can monitor the creation of patient’s PII objects. Once created, the advice can acquire the patient’s identity and use it to generate the specific purpose object with assistance of the patient preference manager. As a result, this new aspect can then examine the fetched records and perform the purpose matching check for determining whether to filter out a particular record or not.

Now, let us resume the discussion of the issue identifiability when retrieving multiple healthcare records. It seems hopeless for the situation that individual’s would not contribute personal data unless the identifiability of “the dataset” satisfies one’s preference first. However, multiparty private computation (MPC) might be applied in this situation\textsuperscript{21,22} because the value of the identifiability of a dataset can be privately computed without reveal any private information pertinent to an individual. For such a situation, we think that we can apply MPC to compute the identifiability of the dataset to retrieve before applying our scheme of privacy aspect.

5. Related Work

Access control enforcement is necessary to protect personal privacy. Role-based access control (RBAC)\textsuperscript{18} is the most often cited guiding principle for application-level security, and is also widely used in the healthcare domain, including HL7\textsuperscript{23}. As we move towards to a privacy-valuing society, purposes of use have been identified as a key ingredient in privacy protection that is missing from traditional access control policies\textsuperscript{8}. Hence, recent standards on security and privacy in many domains all include purposes of use as part of the privacy policy specification, such as P3P\textsuperscript{14}, XACML 2.0\textsuperscript{16}, and HL7 security and privacy vocabulary project\textsuperscript{13}. In the past few years, there have been many architectural proposals published for enforcing purpose-based privacy rules in enterprise applications\textsuperscript{24–26}. Our formulation of privacy-a-
126 Supporting Patients’ Privacy Preferences Using Aspects

ware access control rules are inspired by the works of EPAL as described in 15) and 24). However, both of them emphasize more on the comprehensive features of the proposed architectures rather than practical issues of migrating existing applications to support privacy enforcement, as we do. Neither do they use aspect technology. In the healthcare domain, among other projects, the Hippocratic database project of IBM27 is a notable work that provides database-level protection of the privacy of data it manages. Researchers at IBM have applied it to support privacy policy enforcement in many areas, including the exchange of electronic health records among regional healthcare entities28.

The recent work of Berghe and Schunter29 also proposes to use aspect technology for privacy enforcement. The so-called privacy injector scheme focuses on using aspects to associate privacy preference with personal data and monitor the input and output of PII data in an enterprise application. Their scheme did not include access control enforcement. By contrast, our aspects are more versatile. They can be made to be privacy-aware access control aspects which can be applied to enforce access control as well as privacy rules.

6. Discussion

Privacy protection is becoming an important issue in the development of healthcare information systems. The principle of respecting patients’ privacy preferences has posed significant challenges for building systems that can quickly reflect patients’ preferences regarding the use of their personal information. The traditional design of access control mechanisms hardly includes the requirement of supporting patients’ preferences. In this paper, we have presented an aspect-based preference management framework that collects and manages patients’ preferences independently yet can integrate with the underlying HIS to support patients’ privacy preferences effectively. The proposed mechanisms are loosely coupled with the underlying system. It is thus easy to adapt them and employ them for migrating existing systems to support patients’ privacy preferences.

Besides, our framework has two other extensibility features. First, the patients’ preference manager can be easily adapted for handling the “sticky policy”26 required for cross-organization exchange of personal health information. The idea is quite straightforward. As the patients’ preference manager works largely independent of the other systems, it does not concern the real source of the preferences. It could be derived from the consent forms of local patients as well as the privacy policies bundled with the healthcare records transferred from other healthcare organizations. Second, the privacy aspect can also serve as a mechanism for enforcing stronger access control rules, or even be the access control enforcement tool itself. As described earlier, a privacy rule may incur a stronger condition to check than the existing one in an access control rule. Our privacy aspect can enforce such conditions when given the needed context information.

For future work, we shall investigate three directions of extensions. First, since obligations are also an essential part of comprehensive privacy enforcement, we shall proceed to devise mechanisms for managing privacy obligations in our framework. Basically, as proposed in 30), it will involve an implementation scheme for associating privacy obligations with personal data and provide an event service which periodically scans pending privacy obligations and fulfills qualified ones.

Second, we plan to investigate the support of dynamic adjustment of conditions in privacy-enhanced access control rules. In some situations, it is likely that certain access control rules are subject to frequent changes. In our current approach, for those changes to be effective, we have to modify the involved access control aspects, re-compile
them, stop the application, and re-weave the aspects into the application. If this kind of service interruption is not acceptable, we must provide some kind of dynamic adjustment mechanism which can support replacement of the enforcing code while allowing the application running as usual. We plan to utilize the dynamic class loading facility of the Java virtual machine to provide such a mechanism.

Finally, we shall also look into the issue of conflict resolution when enforcing patients’ privacy protection. Ideally, all privacy rules and patients’ preferences should be consistent. However, in practice, this is seldom the case. There can be conflicts between the purpose of the access request and the combination of existing purposes and privacy rules of PII data the user grant to access. Some kind of conflict solution service may be needed to resolve them. On the other hand, all privacy enforcement frameworks must handle such conflicts, regardless of their implementation scheme. Hence we think we can study existing solutions, and choose and adapt the ones that suit our framework.

Acknowledgement

Part of this work is done during the first author’s visit to the Institute of Information Science, Academia Sinica.

References

8) Massacci F, Zannone N. Privacy is Linking Permission to Purpose, Lecture Notes in Computer Science Vol. 3957, Springer Berlin/Heidelberg.
14) Platform for Privacy Preferences (P3P) Project.


