PRESERVING PRIVACY IN WEB-BASED E-HEALTH SYSTEMS

Yuan Hong

Concordia University, Montreal, QC, Canada

y_hon@cs.concordia.ca

A thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montréal, Québec, Canada

January 2012

Concordia University, Montreal, QC, Canada

y_hon@cs.concordia.ca, 2012

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Yuan Hong**

Concordia University, Montreal, QC, Canada

y_hon@cs.concordia.ca

Entitled: **Preserving Privacy in Web-based e-Health Systems**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Chair

_____ Examiner

_____ Examiner

_____ Examiner

_____ Supervisor

_____ Co-supervisor

Approved _____

Chair of Department or Graduate Program Director

Dr. Nabil Esmail, Dean

Faculty of Engineering and Computer Science

Abstract

Preserving Privacy in Web-based e-Health Systems

Yuan Hong

Concordia University, Montreal, QC, Canada

y_hon@cs.concordia.ca

Safeguarding patients' private information is one of the most challenging issues in the design and implementation of modern e-Health systems. A patient's consent should usually be obtained before his/her private information can be disclosed. Recent advances in Hippocratic Databases (HDB) show a promising direction towards the enforcement of privacy policies in e-Health systems. With HDB, patients need to specify their privacy preferences about what data to be disclosed to which recipients for what purposes. However, this may become a daunting task in a complicated application that involves potentially a large number of combinations of data recipients, purposes, and granularities of data. This thesis tackles issues in applying the HDB design to e-Health systems. More specifically, we design an architecture for integrating APPEL preferences with HDB, extend the original HDB design to support fine-grained privacy authorizations demanded by patients and adapt the design to a multi-dimensional model; we propose a series of methods for patients to more conveniently specify their privacy preferences based on the hierarchies naturally existing

in each dimension of a privacy preference, define meta-policies to resolve potential conflicts between preferences specified over time, and discuss how to represent such preferences with a snow-flake schema in backend databases. Finally, we illustrate the implementation issues and justify our designs with experimental results.

Acknowledgments

I would like to appreciate lots of people who provide help for this thesis.

First of all, I would like to express my sincerely thanks to my supervisors, Dr. Rachida Dssouli and Dr. Lingyu Wang, for their help, guidance and advices to my study at Concordia University, especially to the research work.

In addition, I would thank all group members in the e-Health research. We work together and implement the system well as a team.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Electronic healthcare systems are playing a critical role in today's medical organizations. How to safeguard patients' private information is an important and challenging issue faced by the designer and administrator of e-Health systems. The privacy issue is critical to such systems because most medical data are about individual patients and highly sensitive [4]. Inappropriate disclosures of those data cause privacy breaches to patients, which in turn lead to serious legal and financial consequences to the organization. At the same time, the privacy issue is particularly challenging in e-Health systems due to the usually complex design and implementation of such systems. There exist standards and solutions for addressing the privacy issue in general-purpose applications. The Platform for Privacy Preferences (P3P) developed by the World Wide Web Consortium (W3C) allows users and websites to declare privacy preferences and policies in a machine-readable format [11]. On the other hand, the Hippocratic Databases is a framework for enforcing privacy policies based

on database technologies [5]. The integration of P3P and HDB is a natural solution for preserving privacy in e-Health systems, which forms the basis of my research work.

Meanwhile, the privacy issue in online systems has drawn considerable attentions lately. Users may abandon an electronic service if he/she feels being requested for too much private information. This fact calls for a more flexible way for users to specify their privacy preferences rather than simply accepting or rejecting a fixed set of policies stated by the service provider. For example, in an e-Health system, patients should be allowed to control aspects like *"which parts of their personal data will be disclosed"*, *"to whom the data will be disclosed"*, and *"for what purposes the data will be disclosed"* according to privacy legislation and regulations [1]. However, specifying such preferences becomes a daunting task in a large and complicated application where personal data may exist at different granularity levels, and such data may potentially be disclosed to a large number of recipients and for many different purposes.

In this thesis, I tackle several issues around the integration of P3P and HDB technologies in e-Health systems [15][30][16][29], and propose a hierarchical approach to address these issues.

- First, I design an architecture for integrating P3P preferences with HDB policies. Patients' preferences specified in APPEL (P3P's Language for Privacy Preferences) are mapped to privacy metadata tables stored in HDB. Doctors requesting for private data are authorized against the privacy metadata.

- Second, we study how to support fine-grained privacy authorizations in HDB.

The attribute-level access control turns out to be insufficient for e-Health systems. I provide a solution based on a redesigned schema.

- Third, I extend HDB to support the multi-dimensional model. The original design of HDB based on relational database model is not suitable for multidimensional models used in medical data warehouses.

- Moreover, the hierarchical approach is based on the observation that hierarchies naturally exist in each of the three dimensions of privacy preferences, namely, private data, recipients and purposes. Specifically, private data can be disclosed in different levels of generalization, to recipients with different privileges, and for purposes dominating each other. I propose a series of methods for patients to more easily specify their privacy preferences by taking advantage of the hierarchies. We then study meta-policies for resolving potential conflicts that may arise between preferences specified over time.

- Meanwhile, we discuss how to encode the specified privacy preferences in backend databases and provide a snowflake schema-based design for simplifying the representation of hierarchical authorizations.

- Finally, implementation issues and some experimental results are described.

The rest of this thesis is organized as follows: Chapter 2 reviews the literature; Chapter 3 discusses issues in multi-dimensional database schema using Hippocratic Databases; Chapter 4 proposes a hierarchical approach to the specification of privacy preferences and the related issues of such an approach; Chapter 5 illustrates

implementation issues. Finally, Chapter 6 concludes the thesis.

Chapter 2

LITERATURE REVIEW

In this chapter, necessary background related to my research is introduced in Section 2.1. Related work for my research topic is described in Section 2.2.

2.1 Background

2.1.1 Service-Oriented Architecture

A Service-oriented Architecture (SOA), is a software architecture that uses loosely coupled software services to support the requirements of business processes and software users. Resources on a network in an SOA environment are made available as independent services that can be accessed without knowledge of their underlying platform implementation. SOA can be implemented using one of the protocols (REST, RPC, DCOM, CORBA and Web services) and, for example, might use a file system mechanism to communicate data to a defined interface specification between processes

conforming to the SOA concept. SOA can also be regarded as a style of information system architecture that enables the creation of applications that are built by combining loosely coupled and interoperable services. These services interoperate based on a formal definition (e.g. WSDL) that is independent of the underlying platform and programming language.

The main drivers for SOA adoption are that it links computational resources and promotes their reuse. Enterprise architects believe that SOA can help businesses respond more quickly and cost-effectively to changing market conditions. This style of architecture promotes reuse at the macro level (service) rather than micro level (objects). It can also simplify interconnection to - and usage of - existing IT (legacy) assets. Hereby, according to the macro viewpoint of an object oriented system, service level partition would make the resources much more optimized.

2.1.2 HIPAA

After too many anguished victims let their issues be known on the health privacy protection, the lawmakers finally noticed. In 1996 the US Congress addressed these issues head on, and acted to pass a landmark law to eliminate all those problems by setting specific mandates in all aspects of the transactions between healthcare companies, providers, and carriers. This law is known as Public Law 104-191, the Kennedy-Kassenbaum Bill, more popularly known as the Health Insurance Portability and Accountability Act (HIPAA). [32]

Main significations of HIPAA are:

- Patient's Access to Information

  In order to eliminate the potential problem and errors due to unavailable records, the law states that patients can access their information any time and in a standard format. This information should be immediately available to the patient and, on request, parties acting on their behalf, such as their new doctors and other healthcare professionals.

- Standardized Information Exchange

  HIPAA also mandates that the information related to health insurance must be exchanged in a standard, predefined way.

- Privacy of Information

  HIPAA places a great deal of emphasis on the issue that affects all of us in some form or other: privacy. Medical records are considered private and should be protected, just like any other tangible property, such as money.

The law mandates that organizations must establish a clear security policy that can be verifiable and, more importantly, auditable.

2.1.3 Multi-Dimensional Data Warehouse/OLAP

A data warehouse is the main repository of an organization's historical data. The most important factor leading to the use of data warehouse is that data analysts can perform complex queries and analysis on the information without slowing down the operating systems.

Data warehouses are the basis of important analytical applications, such as Online Analytical Processing (OLAP). OLAP is an approach to quickly providing answers to analytical queries that are multi-dimensional in nature. OLAP is part of a broader category of business intelligence, which also includes extract, transform and load (ETL), relational reporting and data mining. The typical applications of OLAP are in business reporting for sales, marketing, management reporting, business process management (BPM), budgeting and forecasting, financial reporting and similar areas.

Data warehouses configured for OLAP employ a multidimensional data model, allowing for complex analytical and ad-hoc queries with a rapid execution time. N. Pendse has suggested an alternative and perhaps more descriptive term to describe the concept of OLAP: Fast Analysis of Shared Multidimensional Information (FASMI) [33]. They borrow aspects of navigational databases and hierarchical databases that are speedier than their relational kin.

2.1.4 Hippocratic Databases

The Hippocratic Database ("HDB") is a set of technologies that manages disclosure of electronic health records in compliance with data protection laws without impeding the legitimate flow of information. HDB's active enforcement component limits disclosure of personal health information at a fine-grained level in strict accordance with enterprise policies, legal regulations, and individual patient choices. These technologies [4] include (1) active enforcement of fine-grained data disclosure policies, (2) efficient auditing of past database access to verify compliance with policies, (3)

privacy-preserving data mining, (4) de-identification of personal data using an optimal method of k-anonymity, and (5) secure information sharing among autonomous data sources.

Hippocratic databases include a group of technologies for securing electronic health records [4], such as privacy metadata, compliance auditing [2], privacy preserving data mining [7], optimal k-anonymization [9] .etc.

The privacy metadata in the Strawman architecture of Hippocratic database [5] use *Purpose* as a special identical attribute for authorizations in each table. The purpose will be used for authorizations and policies in other two tables of the metadata called Privacy-Authorization Table (PAT) and Privacy-Policies Table (PPT).

In PAT, four attributes are used for privacy authorization:

- Purpose: the purposes for the authorized users using the private data
- Table: the table where the private attribute belongs to
- Attribute: the private attribute in the original database schema
- Authorized-users: the users authorized to access the data

Meanwhile, in PPT, there exist five attributes for policy recording:

- Purpose: the same as in PAT
- Table: the same as in PAT
- Attribute: the same as in PAT

- External-recipients: the actors to whom the data will be disclosed
- Retention: the period during which the data item should be maintained

The privacy metadata in the architecture will be used to store corresponding special data for attribute access control.

2.1.5 P3P and APPEL

The Platform for Privacy Preferences (P3P) is a standard for assisting web applications to collect users' data and corresponding disclosure elements in a machine-readable XML format, and the specification defines the base data schema for data to be collected together with a standard set of uses, recipients, data categories and the association of polices and websites and so on.

In the P3P specification [13], two types of standards are defined: Privacy Policies which describes how to use the collected personal data by the websites (enterprises or medical organizations) and Privacy Preferences which will be undergone a conformance checking against the policies when recipients plan to access data relevant to the preferences.

P3P has provided a standard language to express users' preferences on their personal data, which is A P3P Preference Exchange Language (APPEL) [12]. APPEL files specify which data to be disclosed, with what purpose, and for whom, and it is not designed to be read by end users but for applications which request policies to use the data. The architectures for preferences negotiation and polices checking will

be presented later in this thesis.

Privacy preferences will be expressed in APPEL rules [12]. Users can describe their preferences in a set of preference rules with APPEL. When recipients request the system to access the disclosed data under the definition of some policies, whether to accept the request for data access will be determined by those preferences rules in APPEL and the requested policies in P3P. Since the set of rules are bound to those policies, the contents of the APPEL specifications include some elements of the P3P policies as well. There are two components existing in a rule: Rule Behavior which specifies the triggered actions once a rule takes effect and Rule Body which provides the pattern which is matched against a policy. For the behaviors, when the requested policy conforms to the preferences, the request will be acceptable and the behavior can be valued *"Request"* to accept the request policy, but if the requested policy doesn't comply with the preferences, the request will be rejected and the behavior can be specified *"Block"* to produce such actions. The requests could be somewhat acceptable if the behavior is *"Limited"*.

The connective attribute within APPEL could define some logical operations for the preferences. There are six APPEL connectives: or, and, non-or, non-and, or-exact, and-exact. Two unusual connectives: or-exact and and-exact should be indicated. Or-exact matches if at least one of the contained preference expressions exist in the policy and no other preference expressions contained in that part of the policy, while and-exact matches if all of the contained preference expressions exist at the correct positions in the policy and no other preference expressions included in that part of

the policy.

2.1.6 Fine-Grained Access Control

Privacy legislation, such as the Health Insurance Portability and Accountability Act (HIPAA) [1], usually requires accesses to be granted to only users with appropriate privileges. A typical requirement is that a patient's private data can only be accessed by authorized persons for intended purposes during a given time period. For example, a doctor can only access data of those patients who are under the doctor's treatment [1]. Such fine-grained access control (FGAC) is traditionally handled at the application layer using filtering predicates to modify database queries. This approach is not scalable for complex applications and its security relies on the correct coding of application developers who are usually not familiar with security requirements [27]. In case of third party canned applications, the approach becomes infeasible since the source code is out of the control of the medical organization.

A common approach to FGAC supported by most modern database management systems is the view-based security. That is to allow users to access only certain predefined views instead of the tables. For example, Oracle's implementation of FGAC, known as Row Level Security (RLS) or Virtual Private Database (VPD), allows policies to be attached to tables and triggered by accesses [27]; Sybase's similar solution, named the Row Level Access Control [21], allows users to define access rules that apply restrictions when retrieving data from the system; Microsoft SQL Server primarily supports traditional view-based access control with an optional

feature called row level permissions usable only with table hierarchies [20]; IBM DB2 supports FGAC through a combined mechanism based on views, triggers and special registers [18].

2.2 Related Work

E-Health systems have received significant attentions these years. Over thousands of health care sites exist online, and a large number of them are web-based. A survey of research topics and trends on e-Health systems can be found in [24]. It reported that most of the research is going on the field of Clinical Information Management (CIM) and Electronic Patient Record (EPR), and research on Consumer Health Informatics (CHI) and Bio Medical Cognitive Science (BCS) are required to be developed in the future, while efforts on Telemedicine (TEL) and Medical Language Processing (MLP) are considerable. Protecting patients' privacy is a mandatory requirement in most e-Health systems according to privacy legislation and regulations, such as the Health Insurance Portability and Accountability Act (HIPAA) [1]. Threats to patients' privacy may arise from many aspects of a medical organization. For example, published medical data can lead to attacks on patients' privacy even though the data are sanitized. The concept of k-anonymity requires identifying attributes to be generalized such that any real-world individual can be linked to at least k records in the published data, which is considered a tolerable privacy threat [36]. Privacy threats within a medical organization may come from unauthorized accesses

to sensitive data. A basic requirement found in most privacy regulations is that accesses to patient records should only be granted to users with appropriate privileges, for intended purposes, and during a given time period [1]. Such a requirement for fine-grained access control (FGAC) can be handled by the application layer or by database systems through view-based security. For example in subsection 2.1.6, Oracle's implementation of FGAC, known as Virtual Private Database (VPD), allows policies to be attached to tables and triggered by accesses [27]. Other popular commercial products like Sybase, Microsoft SQL Server, and IBM DB2, all have different degree of support for FGAC.

As mentioned in Section 2.1, the Platform for Privacy Preferences (P3P) is a standard for encoding a user's privacy preferences and an organization's privacy policies in a machine-readable format, such that a user's browser can interact with the organization's website to determine whether the former's privacy preferences matches the latter's privacy policies [11]. P3P provides a standard language for specifying privacy preferences about disclosing private data, namely, A P3P Preference Exchange Language (APPEL) [12]. APPEL developed by the World Wide Web Consortium (W3C) [12] allows a user's privacy preferences to be specified in a machine-readable format. Tools exist for creating APPEL specifications, such as the Java-based editor developed by the European Union's Joint Research Center (JRC) [10]. Using the tool, an APPEL rule can be created either by selecting from predefined rules or by using the advanced mode that allows for creating new rules. Our work can certainly employ APPEL as a standard interface between the client and server, although we

focus on how to simplify a specification using hierarchies. [37] presented a software to query a set of web sites for P3P policies, check the validity of each policy, and analyze the information practices it describes. Meanwhile, totally 588 P3P-enabled web sites have been analyzed by such a software and they presented the first major analysis of the data practices of P3P-enabled web sites. Furthermore, [3] discussed how to implement P3P, which enables web users to gain control over the private information using database technology. Finally, [23] proposed a privacy policy of P3P by which current relational database management systems can be transformed into their privacy-preserving equivalents.

At the backend, Hippocratic Databases (HDB) is designed for preserving privacy in database applications [5], which is a promising solution to managing privacy policies and preferences using database technology [4] [5]. Privacy-authorization tables in an HDB encode privacy preferences using attributes named "Authorized Users" or "Purpose". The active enforcement ("AE") is a framework for deploying an HDB [19], which supports a preference negotiation stage that ensures any access to private data adheres to privacy policies. Our implementation is based on HDB and our hierarchical approach complements the existing negotiation stage found in the HDB AE. Furthermore, some other research also focus on the extension of Hippocratic Database model, such as extending HDB for XML databases with tree-like hierarchies [28] compared to the existing HDB privacy metadata which integrates privacy protection into relational databases. The database schemata and datasets in our implementation are multi-dimensional based. Related to data warehouse/OLAP,

[38] proposed an effective solution to preserve the confidential information while still providing range query accuracy in OLAP data cube. [25] defined a security model for data warehouse which describes security constraints for roles in the data warehouse.

In [31], the authors propose a flexible framework for preference negotiation between customers and enterprises that adopt different processes for the same service. The framework can help users to use the service with minimal disclosures of their private data. In [22], a hierarchical approach is presented to pose the least number of questions to users about their privacy preferences. This can be considered as "asking" users for their privacy preferences, whereas my approach lets users to "tell" the preferences. In 1996, Anderson proposed a security policy model for clinical information system [8]. The model applies traditional confidentiality and integrity models, such as the BLP mode and Clark-Wilson model, to medical data. In contrast to this high-level policy model, chapter 4 of this thesis will address a specific issue in privacy preference specification.

Chapter 3

SCHEMA DESIGN FOR HIPPOCRATIC DATABASES

In this chapter, the service-oriented system architecture and its components for privacy protection will be introduced in Section 3.1. Consequently, Section 3.2 will describe how to integrate finer-grained authorizations using Hippocratic databases and how to establish a multi-dimensional HDB schema designed for Electronic Patient Records (EPR) with fine-grained authorizations support.

3.1 System Architecture

3.1.1 Service-Oriented System Architecture

Our e-Health system includes a portal and various services behind the portal. We adopt a service-oriented architecture for the e-health portal. Traditional designs of

software system usually have difficulties in meeting the functional requirements. The interoperability among heterogeneous components requires a complicated integration process, which then implies unacceptable downtime and efforts for the integration. In contrast, the service oriented architecture (SOA) exposes resources of each software component as standard-conforming services that can be accessed without understanding the underlying implementation details.

The main components of our design are portals interconnected via the Web Services for Remote Portlets (WSRP) protocol. Each portal is a Web-based application that provides users a unified interface to all the services provided by multiple organizations. The portal allows its users to personalize desired content through creating customized webpage, namely, portal interface. Each portal interface also plays the role of a content aggregator by including a collection of related services. Navigation elements inside each portal interface allow users to easily navigate among different collections of services just like surfing the Web. In our design, all client-side processing is supported either by the built-in functionalities of a standard Web browser or through applets that are automatically downloaded and installed into the browser. Users are not required to possess sophisticated computer skills in order to use services provided by the portal.

More details of the proposed architecture are depicted in Figure 1. The e-Health portal comprises the presentation layer of the architecture, which is decoupled from the implementation layer at service providers to allow independent development of

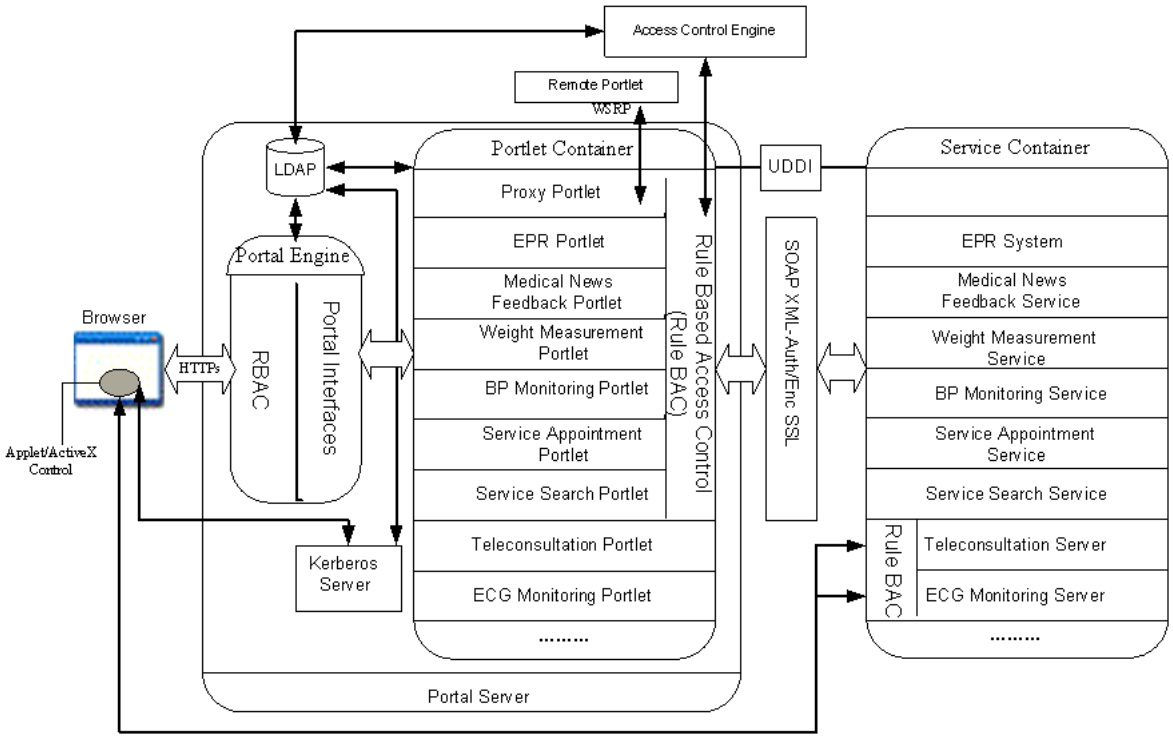

Figure 1: A Service-Oriented Architecture for e-Health Portal

services and to ease the integration of such services. The separation of portal interfaces from portlets allows portal administrators to easily customize the source of services using a content management system. Interoperability and extensibility are both inherent to the architecture because the portlets communicate with backend services through standard Simple Object Access Protocol (SOAP). As an exception, real-time services are allowed to bypass the portal since they usually demand better performance than SOAP can provide. To access a service, a user connects to the portal server via a standard Web browser. According to the user's personalized settings, a portal interface is displayed with a collection of portlets inside it. When the user clicks on a button encapsulated in a portlet, the corresponding action of the service will be performed at backend service providers (which may be governed by a remote portal). For real-time services that bypass the portal, an applet downloaded to the browser will perform the required actions on behalf of the users.

Meanwhile, the privacy protection components include preference negotiation, Electronic Patient Record (EPR) services for data users, database and metadata schemata and so on. In the whole e-Health system, service container stores some EPR services that need to access the private data in the backend database. Patients make preferences through the process of preference negotiation which is integrated into the web interface of the architecture (portlets), while the private data and metadata schemata exist separately from the components in Figure 1. At the time users (such as doctors) want to access those data by some EPR services, our privacy protection engine will be triggered and the access control supported by privacy metadata

will be invoked to limit users in accessing those privacy. The following two subsections will introduce the details of the privacy protection components within a specific architecture.

3.1.2 Preference Negotiation and Data Access

The architecture of the privacy-protection subsystem of an e-Health system is illustrated in Figure 2. We consider two types of users accessing the e-Health system. First, patients state their opt-in and opt-out preferences through a web interface. Second, doctors need to access the personal information of their patients for specific purposes (such as *"Treatment"* and *"Prescription"*). Notice here the patient and doctor only refer to their roles in either providing or requesting the private data. Other users of the e-Health system, such as a nurse, may be considered as a patient, a doctor, or both depending on their roles with respect to the private data.

In Figure 2, patients specify their preferences about disclosing private data in the APPEL language through a web interface. The preferences are checked against the mandatory part of P3P policies for conformance between the client's browser and the web server (this can be implemented in either a client-centric or server-centric way [6] [3]). If the preferences match the policies, the preferences are mapped to and stored in the attributes and records of privacy metadata tables in backend HDB for later references. When doctors request for accesses to private data, the application will provide authentication credentials and associated purposes together with queries. Based on the purposes and the requested resources, such as records and

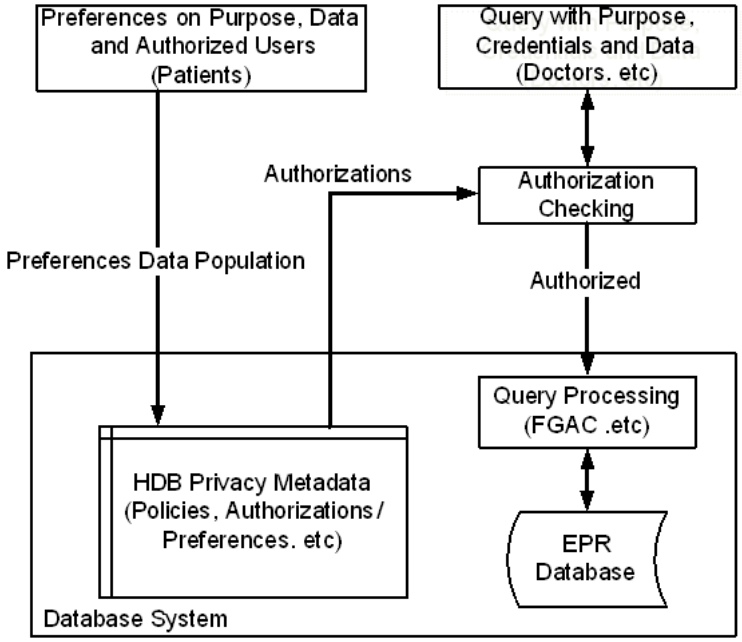

Figure 2: Preference Negotiation and Data Access

attributes, the system checks corresponding metadata and determines whether the doctor is a legitimate recipient of the requested private data. If the doctor haven't been authorized before the checking, the private data would be prohibited for the services requested by the doctor.

3.1.3 Electronic Patient Records Database Schema

Our e-Health system adopts web interfaces (with APPEL integrated) to specify preferences and employs privacy metadata of Hippocratic databases to record preferences for patients. The database system includes schemata for privacy metadata and Electronic Patient Record (EPR) private data, which forms one of the most important concept in this research. Since EPR data will be used for many purposes, such as

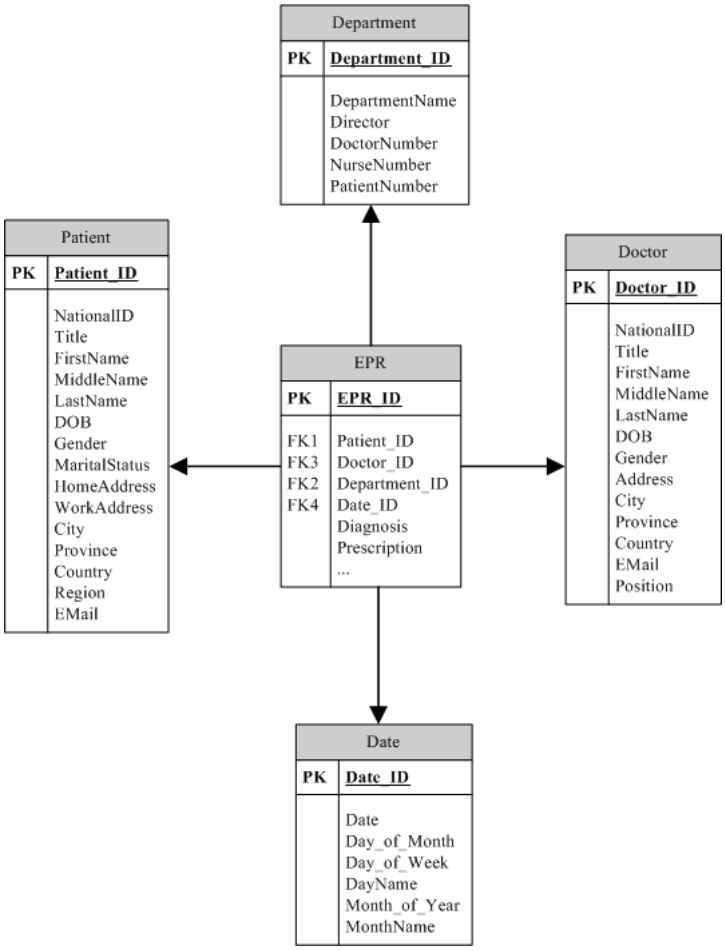

Figure 3: Star Schema for Electronic Patient Records

"Research" or *"Statistics"*, an analytical data warehouse may be required to improve the performance of analytical queries. My research focuses on multi-dimensional medical records and studies how to preserve privacy in such a model. Therefore, a star schema for electronic patient records is given in Figure 3 based on the requirements and characteristics of private medical data warehouses.

As shown in Figure 3, a fact table (EPR) records the dimensional attributes linked to other four dimensional tables, measurement attributes such as *"Diagnosis"*,

"Prescription" .etc, and some other degenerate attributes. Besides the fact table, other dimensional tables provide the specific information of Patient, Doctor, Date and Department for analytical usage. For demo purpose, we have shown only a part of the attributes in each table. In reality, other useful information may be required. Some tables in this simple schema will be referred to in later paragraphs.

3.2 Fine-Grained Authorization in HDB

R. Agrawal et. al proposed Hippocratic databases for preserving privacy using privacy metadata[5]. Table 1 shows schemata of the original HDB design when applied to an e-Health system. Table *Patient* and *EPR* store private data about patients' information and electronic records. Table Privacy-Authorization records which users are authorized to access each combination of table, attribute and purpose (we shall only include a few attributes for simplicity). In this example, attribute *"Diagnosis"* of table *EPR* may be accessible to doctors for treatment purpose and to analysts for statistics purpose, whereas attribute *"Age"* of table *Patient* is only for treatment purpose.

However, the above design has a limitation in that it does not support fine-grained authorizations. In a typical e-Commerce website, a transaction will not even begin unless the user's preferences match all of the website's policies. Each attribute can thus have a fixed set of authorized users no matter in which record it appears. This all-or-none approach is not suitable for medical organizations where, for example,

Table 1: Schemata of the Original HDB Design

HDB based on Relational Database Schema

Table	**Attributes**
EPR	EPR-ID, Patient, Diagnosis, Purpose
Patient	Patient, Gender, Age, Address, City, Purpose

Authorizations in HDB Privacy Metadata

Table	**Attributes**
Privacy-Authorization	Purpose, Table, Attribute, Authorized-Users

Table *Privacy-Authorization*

Purpose	*Table*	*Attribute*	*Authorized-Users*
Treatment	EPR	Diagnosis	Doctor
Treatment	Patient	Age	Analyst
Statistics	EPR	Diagnosis	Analyst

a patient should not be refused of treatment simply because he/she disagrees to provide age or gender for statistics purposes. Any patient should be allowed to opt out optional privacy policies or opt in with conditions. For example, Bob may decide that his diagnosis information should be disclosed for statistics purposes only if the symptom is *Flu*, and the information should be kept private for all other symptoms. As another example, he may choose to allow or disallow a specific doctor in accessing his diagnosis information for certain symptoms. Another patient Eve may have a completely different preference about these private data. None of those requirements can be represented in table *Privacy-Authorization* in Table 1.

Table 2 shows our redesigned schemata for supporting fine-grained access control when the protected data are recorded in a data warehouse. We add an additional attribute *"Authorization-ID"* to the tables and a foreign key constraint on the attributes from table *EPR* to table *Patient*. In table *EPR*, for symptom *Flu*, Bob is willing to

disclose his diagnosis information for both statistics and treatment purposes and his name for treatment purpose but only to Alice. For the symptom of *Diabetes*, Bob does not want to disclose any information for any purpose. Also for symptom *Flu*, Eve has a preference different from Bob's, which is to disclose the diagnosis data to doctors. In addition, the fine-grained authorization can be used in other tables, such as *Patient*. We can see that both of Bob and Eve authorized the access on attributes (*"Age"*, *"Gender"* .etc) for each tuples in the patient information table.

In summary, our redesigned schemata provide all necessary fine-grained authorization described in the above examples. Other special authorizations will be illustrated later on.

The above design of table *Privacy-Authorization* assumes a closed policy. That is, unless a user explicitly appears in attribute Authorized-Users in table *Privacy-Authorization*, the user will by default be prohibited from accessing the attribute. Such a closed policy [35] is inconvenient when a patient chooses to prohibit certain users while allowing all others to access his/her private data. For example, Bob does not want Alice to see his name but does not care about other doctors accessing that data. It would be prohibitive to explicitly list all other doctors' names in attribute *"Authorized-Users"*.

We address this issue by making attribute Authorized-Users a signed attribute. That is, each user appearing in the attribute is either to be allowed or disallowed, as denoted by a preceding sign + or -. Table 3 shows an example where the second authorization prohibits a specific user Alice from accessing the patient's name while

Table 2: Relational Schemata for Fine-grained Authorizations

HDB Schemata

Table	**Attributes**
EPR	EPR-ID, Patient, Diagnosis, Purpose, Authorization-ID
Patient	Patient, Gender, Age, Address, City, Purpose, Authorization-ID

HDB Privacy Metadata Schema

Table	**Attributes**
Privacy-Authorization	Authorization-ID, Purpose, Table, Attribute, Authorized-Users

Table *EPR*

EPR-ID	*Patient*	*Diagnosis*	*Purpose*	*Authorization-ID*
000001	Bob	Flu	Treatment	1,2
000001	Bob	Flu	Statistics	3
000002	Bob	Diabetes		
000003	Eve	Flu	Treatment	1

Table *Patient*

Patient	*Gender*	*Age*	...	*Purpose*	*Authorization-ID*
Bob	Male	28	...	Treatment	4
Bob	Male	28	...	Research	5,6
Eve	Female	23	...	Treatment	4

Table *Privacy-Authorization*

Authorization-ID	*Purpose*	*Table*	*Attribute*	*Authorized-Users*
1	Treatment	EPR	Diagnosis	Doctor
2	Treatment	EPR	Patient	Analyst
3	Statistics	EPR	Diagnosis	Analyst
4	Treatment	Patient	Age	Doctor
5	Research	Patient	Age	Researcher
6	Research	Patient	Gender	Researcher

Table 3: Signed Attribute *Authorized-Users*

Table *Privacy-Authorization*

Authorization-ID	*Purpose*	*Table*	*Attribute*	*Authorized-Users*
1	Treatment	EPR	Diagnosis	+Doctor
2	Treatment	EPR	Patient	+Doctor, -Alice
3	Treatment	EPR	Patient	+Nurse, +Chris

allowing all other doctors to access the same data. An additional issue here is the potential conflicts between signed values. For example, if Alice is a doctor, then the first signed Value +Doctor will grant Alice the access (since she is a doctor) but the second value -Alice will prohibit this access. The solution is to define meta-policies for resolving conflicts, such as the most-specific-take-precedence meta-policy by which Alice will be prohibited (since Alice is more specific than Doctor).

Furthermore, our mechanism supports the union of authorizations as well as the open policy. Take the authorization 3 in Table 3 as an instance, the information of *"EPR.Patient"* can be disclosed to Nurse and Chris using the same tuple of Table *Privacy-Authorization*. In that example, suppose that Chris is a doctor but not the nurse. However, if a patient makes his/her preference, and he/she thinks that the private data should be disclosed to another doctor, Chris, for the same purpose as well as the preference he/she is making for nurse at the same time. The authorizations for this case can be represented as *"Authorization 3"* (a union) in Table 3.

3.3 Multi-Dimensional Hippocratic Databases

The original design of HDB is based on the relational database model, which is suitable for small to medium size operational applications. However, analytical applications dealing with a large amount of data, such as a medical data warehouse, will typically adopt a multi-dimensional model instead of the relational model [33]. We now extend the schema design introduced in subsection 3.2.1 to the multi-dimensional model. We also make optimizations to reduce the storage requirements of privacy metadata.

A multi-dimensional model such as a star schema, consists of a number of dimensional tables and a fact table, with foreign key constraints linking the two (a simple multi-dimensional EPR database schema has been given in subsection 3.1.3). Table 4 shows a star schema augmented with attributes *"Purpose"* and *"Authorization-ID"*. Table *EPR* is the fact table. It can be observed that the fact table includes much redundancy due to various purposes associated with the same record. Such redundancy may be acceptable in an operational database, but it is usually prohibitive for analytical applications based on a multi-dimensional model where the cardinality of the fact table can easily go beyond millions. This situation will be exasperated when table *EPR* includes attributes of object data types, such as laboratory test measurements in multimedia format (for example, X-ray, CT scan or MRI images).

To avoid the extra storage overhead introduced by multiple purposes, we decompose the fact table through schema normalization. As shown in Table 5, we add

Table 4: HDB in Star Schema

HDB based on Star Schema
with Fine-grained Authorizations

Table	**Attributes**
EPR (fact table)	Patient-ID, Doctor-ID, Date-ID, Diagnosis, , Prescription, Purpose, Authorization-ID
Patient	Patient-ID, Name, Gender, Age, Address, City, Region, Province, Country, Purpose, Authorization-ID
.....	.....

Table *Fact Table EPR*

Patient-ID	*...*	*Diagnosis*	*...*	*Purpose*	*Authorization-ID*
001	...	Flu	...	Treatment	1,2
001	...	Flu	...	Statistics	3
001	...	Flu	...	Prescription	4
001	...	Flu	...	Research	5
001	...	Diabetes	...	Treatment	1,2
...	...	...	...	...	...

attribute *"EPR-ID"* as a primary key to table *EPR* and at the same time remove attributes *"Purpose"* and *"Authorization-ID"* from the table. We then introduce a new table *Purpose-Authorization* to record the purpose and authorizations for each EPR record. We do not normalize the dimensional tables since they typically have a much smaller cardinality than that of the fact table and the redundancy introduced by multiple purposes is thus usually acceptable.

Another issue relevant to a multi-dimensional HDB is that there are two different levels of authorizations for an attribute in the dimensional tables. For example, Bob may want his address to be disclosed to doctors for treatment purpose, which can be represented by an authorization for table Patient and attribute Address. However, Bob may want his address disclosed for symptom *Flu* but not for *AIDS*. This second

Table 5: Schema for Multi-dimensional HDB

Table	**Attributes**
EPR *(fact table)*	EPR-ID, Patient-ID, Doctor-ID, Date-ID Diagnosis, Prescription
Patient	Patient-ID, Name, Gender, Age, Address, City, Region, Province, Country, Purpose, Authorization-ID
Doctor	Doctor-ID, Name, Gender, Age, Position, Address, City, Region, Province, Country, Purpose, Authorization-ID
.....	.....
Purpose-Authorization	EPR-ID, Purpose, Authorization-ID

Table 6: Access Control for Dimensional Tables

Fact Table *EPR*

EPR-ID	*Patient-ID*	*...*	*Diagnosis*	*...*
000001	001	...	Flu	...
000002	001	...	AIDS	...

Table *Patient*

Patient-ID	*Name*	*Gender*	*Address*	*...*
001	Bob	...	...	...

Table *Purpose-Authorization*

EPR-ID	*Purpose*	*Authorization-ID*
000001	Treatment	1

Privacy-Authorization

Authorization-ID	*Purpose*	*Table*	*Attribute*	*Authorized-Users*
1	Treatment	Patient	Address	Doctor

authorization cannot be represented as above because the *"Authorization-ID"* in table *Patient* is not associated with attribute *"Diagnosis"* in the fact table. Our solution is to allow an authorization in the fact table to specify an attribute in the dimensional table. As shown in Table 6, the authorization has its *"Authorization-ID"* appearing in the fact table (that is, table *Purpose-Authorization*) but specifies an attribute *"Address"* in a dimensional table *Patient*.

From the fact table *EPR*, there exist two tuples for the same patient Bob, with different Diagnosis: *Flu* and *AIDS*. Bob doesn't want his address information to be disclosed along with the diagnosis information. In this situation, the table *Privacy-Authorization* in our multi-dimensional HDB provides authorizations for the relationship between fact table tuples and information in dimensional tables. The authorization in table *Privacy-Authorization* (of Table 6) limits the access to *"Patient.Address"* for doctors, simultaneously, the tuples in *Privacy-Authorization* limits the access at a finer-grained level, that is, only *"EPR 000001"* can be disclosed with a diagnosis *Flu*.

Chapter 4

HIERARCHICAL SPECIFICATION OF PRIVACY PREFERENCES

This chapter proposes a hierarchical approach to the specification of privacy preferences. First, Section 4.1 defines hierarchies that are inherent to each dimension of privacy preferences. Section 4.2 then proposes methods to simplify a user's tasks in specifying his/her privacy preferences. Section 4.3 addresses conflict resolution for specified privacy preferences. Finally, Section 4.4 describes the implementation of this hierarchical approach.

4.1 Hierarchies in Privacy Preferences

In a complicated application, such as an e-Health system, we are usually dealing with data organized at different granularity levels, data recipients with different ranks and responsibilities, and purposes that may dominate each other. For example, a patient's address naturally forms a hierarchy: *"Address → City → Province → Country"*. Here each arrow indicates a further level of data generalization.

Two facts about hierarchies are worth noting. First, in terms of privacy, more generalized data are usually less sensitive. For example, a patient may not want his/her exact address to be published but he/she may be less concerned about the state or country. Second, the levels of generalization usually form a partially ordered set (poset)[26]. As depicted in Figure 4, the *"Home-Address"* and *"Work-Address"* may not be comparable in terms of sensitivity, neither do the *"State"* (*"Province"*) and *"Region"*.

Hierarchies are also inherent to the other two dimensions of privacy preferences, namely, recipients of data and purposes. First, recipients of private data in an organization typically have their privileges assigned based on their ranks or positions. For example, Figure 5 shows a simple hierarchy on data recipients in an e-Health system. In order to supervise nurses, a supervisor may need to access the same data accessible to the nurses.

Second, purposes can usually be refined into sub-purposes. As mentioned in [31],

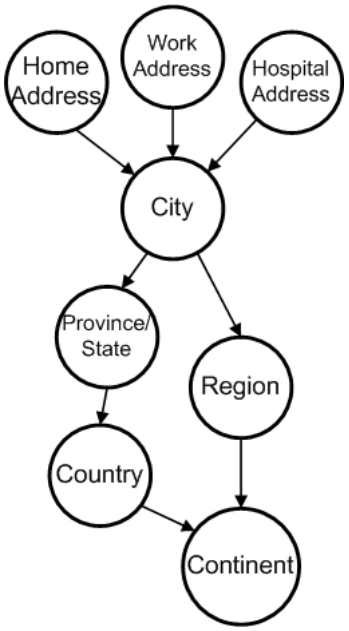

Figure 4: Hierarchy on Private Data

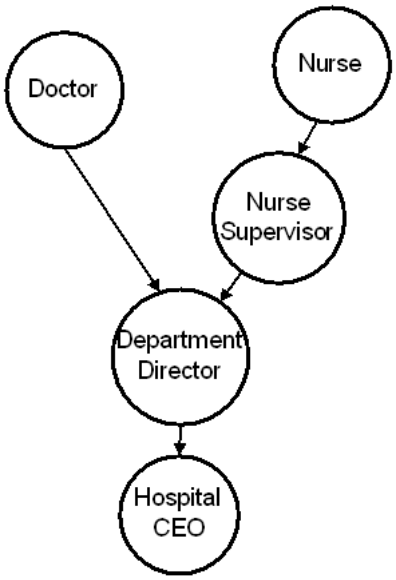

Figure 5: Hierarchy on Data Recipients

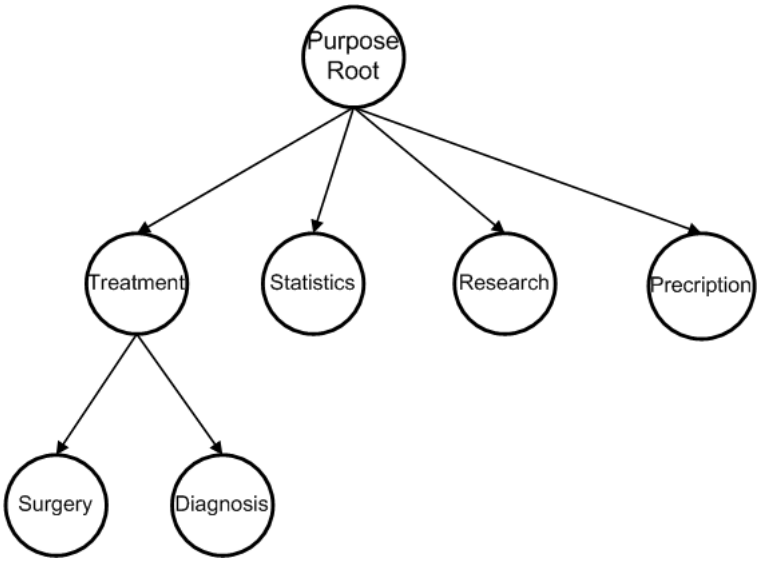

Figure 6: Hierarchy on Purposes

the purpose *"Shipping"* dominates sub-purposes *"Regular-Shipping"* and *"Shipping-by-Courier"*, *"Delivery"* dominates *"Direct-Delivery"* and *"Delivery-by-Post"*, and *"Notification"* dominates *"Notification-by-Fax"* and *"Notification-by-Email"*. In an e-Health system, we consider such purposes: *"Treatment"*, *"Statistics"*, *"Research"*, *"Surgery"*, *"Prescription"*, and *"Diagnosis"*. It is not difficult to see that *"Surgery"* and *"Diagnosis"* can be regarded as the sub-purpose of *"Treatment"*, as shown in Figure 6.

We can define a single hierarchy combining all three dimensions of privacy preferences. However, additional cares need to be taken during such a combination because hierarchies on those three dimensions have different semantics. We approach this issue by taking the point of view of a user specifying his/her privacy preferences. For example, if the user agrees to disclose the address, then likely he/she may not care

about the city, state, or other coarser grained data (we shall address exceptions later on). Similarly, if the user is willing to disclose the data to a nurse, then the nurse's supervisor may also need that access; if the consent is obtained for shipping purpose, then the consent may simply extend to both sub-purposes. That is, the three dimensions should be combined in such a way that the finest-grained data will coincide with recipients with the least privileges and purposes that are not dominated by others, as formally stated in Definition 1.

Definition 1: *Let (D, $\preceq_D$), (R, $\preceq_R$), (P, $\preceq_P$) be posets defined on private data, recipients, and purposes, respectively; the relations $\preceq_D$, $\preceq_R$, and $\preceq_P$ are defined in such a way that $d_1 \preceq_D d_2$ if d_1 is less sensitive than d_2, $r_1 \preceq_R r_2$ if r1 has more privileges than r_2 does, and $p_1 \preceq_P p_2$ if p_1 is a sub-purpose of p_2. We define the privacy preference poset as the product of these three posets $<M, \preceq>$ where $M=D\times R\times P$, and $\preceq$ is the composition of the three relations $\preceq_D$, $\preceq_R$, and $\preceq_P$.*

4.2 Specifying Preferences Using Hierarchies

We now propose methods for specifying privacy preferences. Our methods aim to simplify the specification of preferences by taking advantage of the privacy preference poset given in Definition 1. More specifically, the poset is defined in such a way that the consent on disclosures at a higher level in the poset implies the consent on disclosures at lower levels. Such disclosure inheritance may allow a user to specify a privacy preference without explicitly enumerating all the levels he/she is willing to

disclose. On the other hand, exceptions to disclosure inheritance may certainly exist and users should be allowed to specify such exceptions inside a specification. The specification of exceptions can again employ the hierarchies if necessary.

We consider three options for specifying a preference, with increasing degree of utilization of the disclosure inheritance (notice that other options also exist but are not discussed here). For each option, we show how to generate the effective set of attributes to be disclosed according to the preference, namely, the disclosure set. We shall use the example of a poset on the data dimension as given in Figure 4 for simplicity. Also for simplicity, we denote the poset using an abstract version *(H, $\preceq$)*, as shown in Figure 7, where $H = \{a, b, c, d, e, f, g, h\}$ with a for *"Home Address"*, b for *"Work Address"*, c for *"Hospital Address"*, d for *"City"*, e for *"Province/State"*, f for *"Region"*, g for *"Country"*, and h for *"Continent"*. The relation $\preceq$ remains the same.

4.2.1 Option 1

This option allows a user to explicitly specify either what to disclose or what to be kept private. That is, the disclosure inheritance is not employed at all. Formally, given the privacy preference poset $<M, \preceq>$, the privacy preference specification is simply a set selected by the user $X \subseteq M$. We can then generate the disclosure set as either $D = X$ (if X is to be disclosed) or $D = M - X$ (if X is to be kept private).

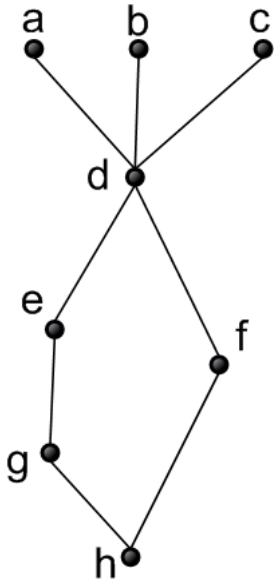

Figure 7: Hasse Diagram for Hierarchy on Private Data

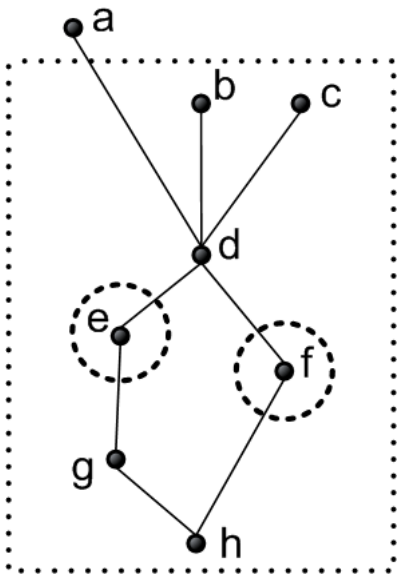

Figure 8: Disclosure Inheritance for Option 2

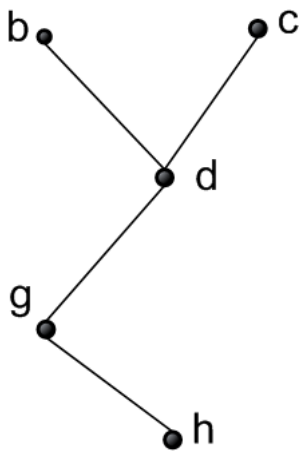

Figure 9: Disclosure Set Example for Option 2

4.2.2 Option 2

The second option allows a user to partially utilize the disclosure inheritance. More specifically, he/she can specify a range in the privacy preference poset $<M, \preceq>$, the preference is specified with an upper bound $X \subseteq M$, a lower bound $Y \subseteq M$, and a set Z to be kept private. We can then generate the disclosure set $D \subseteq M$ as the set $\{d | \exists x \in X, \exists y \in Y, y \leq d \leq x\} - Z$. For example, Figure 8 shows a specification where $X = \{b, c\}$, $Y = \{h\}$, and $Z = \{e, f\}$. The disclosure set is $D = \{b, c, d, g, h\}$, as shown in Figure 9. Notice the relation *(d, g)* is due to transitivity.

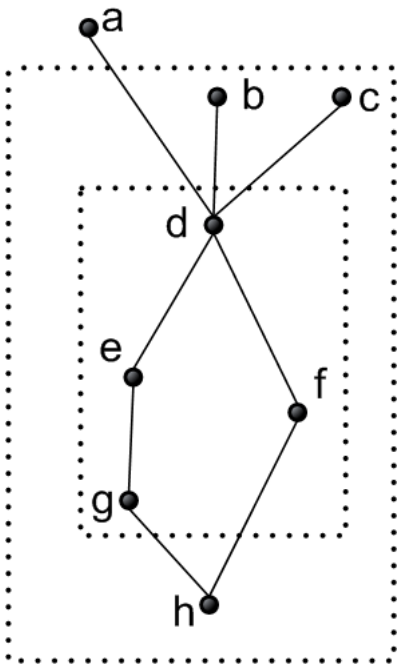

Figure 10: Disclosure Inheritance for Option 3

4.2.3 Option 3

The third option allows a user to fully utilize the disclosure inheritance by specifying ranges in the privacy preference poset both for disclosing and for keeping private. Given the privacy preference poset $<M, \preceq>$, the preference is specified with a pair of upper and lower bounds $X \subseteq M$ and $Y \subseteq M$ for disclosing, and another pair $W \subseteq M$ and $Z \subseteq M$ for keeping private. We can then generate the disclosure set $D \subseteq M$ as $\{d | \exists x \in X, \exists y \in Y, y \leq d \leq x\} - \{p | \exists w \in W, \exists z \in Z, w \leq p \leq z\}$. For example, Figure 10 shows such a specification where $X = \{b, c\}$, $Y = \{h\}$, $W = \{d\}$, and $Z = \{g, f\}$. The disclosure set is $D = \{b, c, h\}$ and is shown in Figure 11.

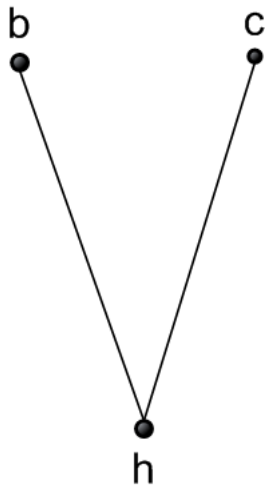

Figure 11: Disclosure Set Example for Option 3

4.3 Conflict Resolution

In my solution, each preference specified by users will be stored in the backend database in the form of its corresponding disclosure set for later reference. However, preferences specified over time may conflict with each other. Prompting the user to resolve the conflicts by choosing a preference over the other is a possible solution. However, it is usually not feasible to completely rely on users for making every decision. Instead, we can provide users general guidelines for resolving a potential conflicts, namely, meta-policies. We consider three representative meta-policies (other meta-policies are also possible), namely, *"Latest Take Precedence"*, *"Disclosure Take Precedence"*, and *"Denial Take Precedence"*.

4.3.1 Latest Take Precedence

This meta-policy allows a user to update his/her preferences. That is, any potential conflict between the new preference and existing preferences will be resolved by overriding the latter with the former. Since we store previous preferences as their corresponding disclosure sets, the positive part of a new specification (that is, what to be disclosed) will not cause conflicts with existing preference. However, the negative part (that is, what to be kept private) may cause a conflict if it overlaps with existing preferences.

More specifically, given the privacy preference poset <M, $\preceq$>, let $D_0 \subseteq M$ be the disclosure set of existing preferences, then the new disclosure set D_1 can be generated according to different options of specification as follows. For the first option, $D_1 = D_0 \bigcup X$ (if X is to be disclosed) or $D_1 = D_0 - X$ (if X is to be kept private). For the second option, $D_1 = D_0 \bigcup \{d | \exists x \in X, \exists y \in Y, y \leq d \leq x\} - Z$. For the third option, $D_1 = D_0 \bigcup \{d | \exists x \in X, \exists y \in Y, y \leq d \leq x\} - \{p | \exists w \in W, \exists z \in Z, w \leq p \leq z\}$. It is worth noting that we cannot simply compute the disclosure set of the new specification, and then union the result with the existing disclosure set. This is due to the simple fact $A \bigcup (B - C) \neq A \bigcup B - C$. For example, for the second option, the union between the existing and new disclosure sets would be $D_0 \bigcup (\{d | \exists x \in X, \exists y \in Y, y \leq d \leq x\} - Z)$, which is different from $D_0 \bigcup \{d | \exists x \in X, \exists y \in Y, y \leq d \leq x\} - Z$.

For example, if the existing disclosure set in Figure 5 is $\{a, b, c, d\}$, and the latest disclosure set is $\{b, c, d, e, f\} - \{c, d\}$, the new disclosure set would be $\{a, b, c, d\} \bigcup \{b, c, d, e, f\} - \{c, d\} = \{a, b, e, f\}$ but not $\{a, b, c, d, e, f\}$.

4.3.2 Disclosure Take Precedence

This meta-policy overrides any conflicting preferences with the consent to disclosure. The meta-policy models the situation where accesses are not audited. In such a case, we must assume that any disclosure may happen immediately after it is consented by the user, so withdrawing such consent is not possible. Since we store previous preferences as their corresponding disclosure sets, any conflict caused by the negative part of a new preference will be ignored. Computing the new disclosure set by this meta-policy is straightforward, because we simply union the new disclosure set with the existing one. More specifically, for the first option, $D_1 = D_0 \bigcup X$ (if X is to be disclosed) or $D_1 = D_0$ (if X is to be kept private). For the second option, $D_1 = D_0 \bigcup (\{d | \exists x \in X, \exists y \in Y, y \leq d \leq x\} - Z)$. For the third option, $D_1 = D_0 \bigcup (\{d | \exists x \in X, \exists y \in Y, y \leq d \leq x\} - \{p | \exists w \in W, \exists z \in Z, w \leq p \leq z\})$.

For example, if the existing disclosure set in Figure 5 is $\{a, b, c, d\}$, and the latest disclosure set is $\{b, c, d, e, f\} - \{c, d\}$, the new disclosure set would be $\{a, b, c, d, e, f\}$.

4.3.3 Denial Take Precedence

This meta-policy is the dual to the *"Disclosure Take Precedence"* meta-policy. That is, a user can disallow a disclosure even if it has previously been consented upon. This meta-policy may be more desirable in the viewpoint of a user. Because we already store previous preferences as disclosure sets, this meta-policy behaves in exactly the same way as the *"Latest Take Precedence"* meta-policy, although they have different semantics. That is, we can generate the new disclosure set by subtracting the negative

part of the new specification from the union of the existing and new disclosure sets.

We have been assuming that preferences are stored in the form of disclosure sets. Some applications may prefer the specification itself to be stored without explicitly computing its corresponding disclosure set. For example, if most preferences involve a large portion of the privacy preference poset, then storing specifications rather than disclosure sets will lead to significant savings in storage (at the expense of more computational overhead for computing the disclosure set at run time). With such an approach, the negative part of existing preferences and that of the new preference can both cause conflicts. Moreover, the *"Denial Take Precedence"* meta-policy will need to be handled differently from the *"Latest Take Precedence"* meta-policy. Specifically, for the *"Latest Take Precedence"* meta-policy, we need to compute disclosure sets in an incremental manner. For example, for the first option, this amounts to iteratively evaluate $D_i = D_{i-1} \bigcup X_i$ (if X_i is to be disclosed) or $D_i = D_{i-1} - X_i$ (if X_i is to be kept private) starting from the first specification to the current one. For *"Denial Take Precedence"* meta-policy, on the other hand, we can simply compute the final disclosure set as $\bigcup D_i - \bigcup X_i$.

4.4 Implementation Option

This section will discuss the implementation of hierarchical authorizations, introduce how to represent those authorizations using privacy metadata of Hippocratic databases, and propose a snowflake metadata schema to record the authorizations,

which is capable of substituting existing metadata when necessary.

4.4.1 Using HDB Metadata

As we have introduced in previous chapters (especially Chapter 3), we can implement fine-grained authorizations based on Hippocratic Databases. HDB privacy metadata is the basis of the hierarchical authorization as well as fine-grained authorization. In early section of this chapter, we illustrated the three dimensions in *Privacy-Authorization Table*, the disclosure set based on hierarchies and conflict resolution. We can use *Privacy-Authorization Table* to record all the authorizations and a group of tuples (authorizations) to represent the disclosure set. We extract this type of disclosure set from metadata schema to resolve conflicts resulting from authorizations executed at different time.

For example, if an authorization is given as the following: $\{City, Province, Country\}$, $\{Nurse, NurseSupervisor\}$, $\{Treatment, Surgery\}$(for the relationships of those elements, refer to Figure 4, 5 and 6), then totally twelve authorizations will be derived, as shown in Table 7.

There are twelve tuples in *Privacy-Authorization Table*. This original HDB solution performs well for fine-grained authorizations after adding an *"Authorization-ID"* column. Nevertheless, we cannot recognize the hierarchies (especially the relationships) in those three dimensions from the given authorizations. Thus, we cannot easily extract existing disclosure sets to compare with latest disclosure set for conflict resolution. For this reason, we will discuss another solution in subsection 4.4.2.

Table 7: Authorization under Original HDB Design

Authorization-ID	*Purpose*	*Table*	*Attribute*	*Authorized-Users*
1	Treatment	Patient	City	Nurse
2	Treatment	Patient	Province	Nurse
3	Treatment	Patient	Country	Nurse
4	Treatment	Patient	City	Nurse Supervisor
5	Treatment	Patient	Province	Nurse Supervisor
6	Treatment	Patient	Country	Nurse Supervisor
7	Surgery	Patient	City	Nurse
8	Surgery	Patient	Province	Nurse
9	Surgery	Patient	Country	Nurse
10	Surgery	Patient	City	Nurse Supervisor
11	Surgery	Patient	Province	Nurse Supervisor
12	Surgery	Patient	Country	Nurse Supervisor

4.4.2 Using Snowflake Metadata

Hierarchies naturally exist in many aspects of an HDB, such as EPR data, authorized users, and purpose. We have proposed a hierarchical approach for simplifying the specification of privacy preferences and authorizations by leveraging such hierarchies. However, how to represent privacy authorizations in an HDB in the presence of such hierarchies is not addressed, and we have discussed the shortcoming of representing hierarchical authorizations under original HDB in section 4.4.1. we now describe a snowflake schema-based design to solve this problem.

Privacy authorizations can be derived from hierarchies on EPR data, authorized users, and purposes. For example, if *"Address"* has been authorized then less sensitive data of *"City"* or *"Country"* are implicitly authorized altogether (exceptions to such inheritance are handled in Section 4.2, and we shall not address it here). In the authorized user dimension, if *"Nurse"* is authorized to access certain EPR data, then

users with higher privileges, such as *"Nurse Supervisor"* will be authorized to access the same data as well. For the purpose dimension, if a purpose *"Treatment"* has been authorized then a more specific purpose (sub-purpose) such as *"Surgery"* will also be authorized. It is worth noting that the inheritance of authorizations is reversed for the purpose dimension in contrast to the other two. For example, *"Country"* dominates *"City"* (in the sense that a country may correspond to many cities) and authorizations on *"City"* imply those on *"Country"*, whereas purpose dominates sub-purpose but authorizations on purpose imply those on sub-purpose.

Neither the original HDB design nor the design in previous sections can provide efficient implementation of such hierarchical authorizations. We provide a solution based on the snowflake schema instead. Table 8 shows an example of our design. For simplicity, we only include the address hierarchy in EPR data. In this schema, all tables are fully normalized, and the hierarchical relationships between attributes are represented as foreign key constraints. The metadata tables include both the authorized user dimension and the purpose dimension.

In the presence of hierarchies, authorizations can be given as any combination of three sets of attributes where each set is from the EPR data, authorized user, and purpose dimension, respectively. Under the original design of HDB, this may result in a large number of authorizations. For example in Table 7.

Table 8: Snowflake Schema for EPR Data and Metadata

Hierarchies in EPR Schema

EPR Table	**Attributes**
Address	Address-ID, Address, City-ID
City	City-ID, City, Province-ID
Province	Province-ID, Province, Country-ID
Country	Country-ID, Country

Hierarchies in Metadata

Metadata Table	**Attributes**
Nurse	Nurse-ID, Nurse-Name, Supervisor-ID
Supervisor	Supervisor-ID, Supervisor-Name
...	...
Sub-Purpose	Sub-Purpose-ID, Sub-Purpose, Purpose-ID
Purpose	Purpose-ID, Purpose

Table 9: Redesigned Schema for Hierarchical Authorizations

Table	**Attributes(Columns)**
Privacy Authorization (fact table)	Authorization-ID, EPR Table, Data-ID, Purpose Table Purpose-ID, Authorized-Users Table, Authorized Users-ID

With our approach, the collection of authorizations in Table 7 can be conveniently represented using a single record. More specifically, we redesign the *Privacy-Authorization Table* as shown in Table 9. In addition to the attribute *"Authorization-ID"* (required for fine-grained authorization as described in Chapter 5), each authorization now includes three pairs of attributes. Each pair indicates the exact elements in the schema is authorized in each of the three dimensions: EPR data, authorized users, and purpose.

Table 10 shows an example where the Data-ID is linked to Address-ID, City-ID, etc. Due to the aforementioned hierarchies, the single record (*"Authorization 1"*) in Table 10 is sufficient for representing the twelve authorizations given in Table 7,

Table 10: Authorization under the Redesigned Schema

Authorization -ID	*EPR Table*	*Data -ID*	*Purpose Table*	*Purpose -ID*	*Authorized Users Table*	*Authorized User-ID*
1	City	All	Purpose	1	Nurse	All
2	Address	1	Sub-Purpose	2	Nurse Supervisor	2

in which *"Purpose 1"* represents *"Treatment"* in Table *Purpose* and the value *"All"* stands for all the disclosed tuples for this elements.

From viewpoint of fine-grained access control, this snowflake metadata can represent not only the hierarchical authorizations, but also the authorization in a single tuple. In Table 10, the values of other three columns: Data-ID, Purpose-ID and Authorized Users-ID can be used to determine authorizations on either none-or-all or exact instances. This is consistent with the fine-grained authorizations introduced in Chapter 3. The fact table *Privacy-Authorization* can record not only the elements to be disclosed (such as City, Nurse in Table 10) but also the exact values of elements in the disclosure sets (such as Data-ID and Authorized User-ID that refer to the fine-grained level data and recipients). As for the disclosure purpose, we must use this fine-grained level of disclosure, because the exact sub-purpose or purpose (such as Surgery or Treatment) are what we need to complete the disclosure inheritance. For example in Table 10, the record *"Authorization 2"* means that *"Address 1"*(and more generalized data) is disclosed to *"Nurse Supervisor 2"* (and higher-positioned recipients) for *"Sub-Purpose 2"*.

Figure 12 illustrated the snowflake metadata schema for the implementation of hierarchical authorizations. For simplicity, only a few attributes in the hierarchies

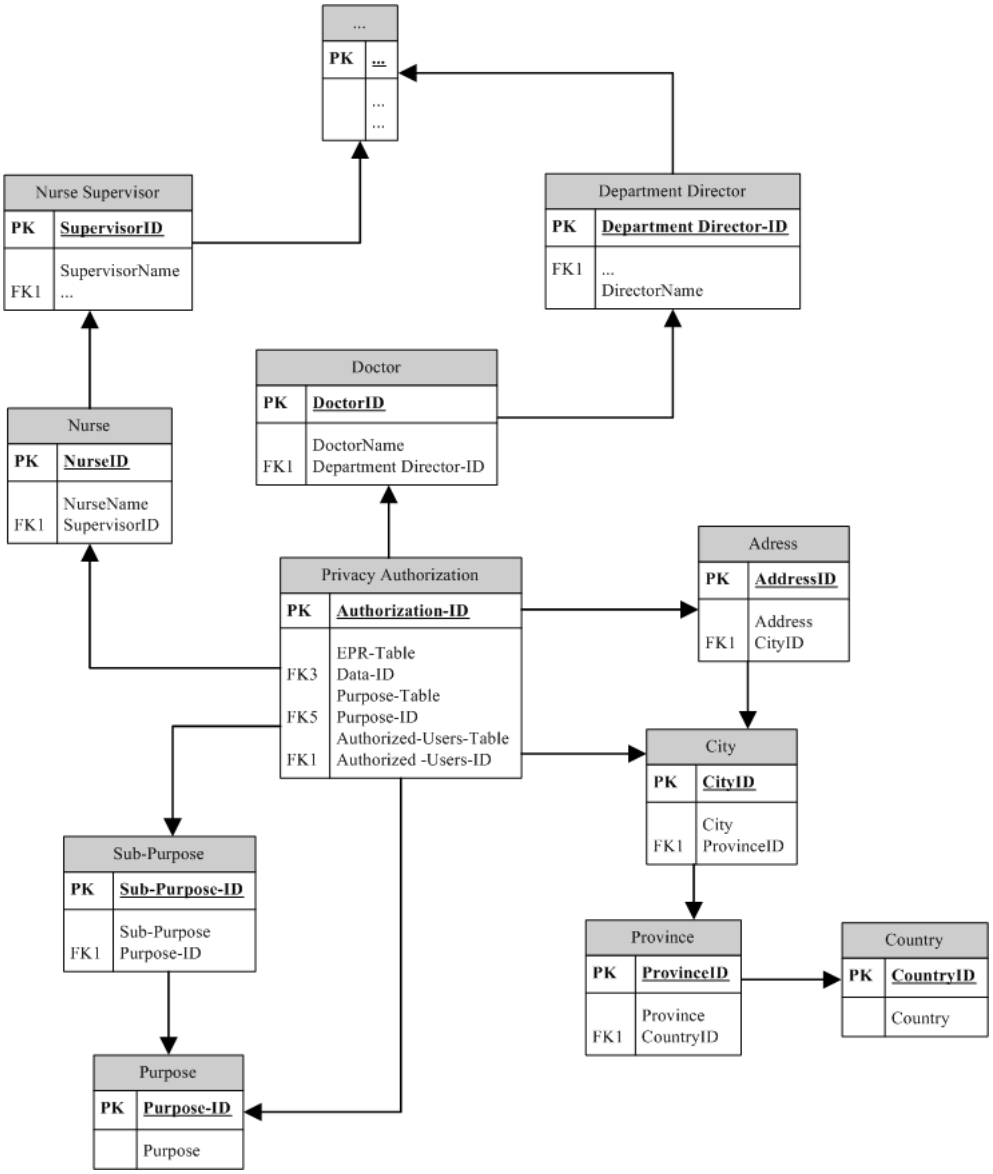

Figure 12: Snowflake Metadata for Hierarchical Authorizations

have been presented. Some facts are worth noting:

- *Privacy-Authorization Table* is the fact table of this schema, and it can be linked to every table of the three dimensions with constraints on three foreign keys. In order to provide better presentation of Figure 12, only Table *Doctor*, *Nurse*, *Address*, *City*, *Sub-Purpose* and *Purpose* have been linked to the fact table. Actually, *"Data-ID"* in the fact table can be linked to *"Address-ID"*, *"City-ID"* .etc; *"Authorized-Users-ID"* can be linked to *"Nurse-ID"*, *"Doctor"* .etc; and *"Purpose-ID"* can be linked to *"Purpose-ID"* and *"Sub-Purpose-ID"*.

- In the fact table, the attributes: *"EPR-table"*, *"Purpose-Table"*, *"Authorized-Users-Table"* have recorded the links from *Privacy-Authorization Table*. Once the names of three tables are recorded in those three columns, respectively, such as *City*, *Nurse* and *Purpose*, all the lower layer tables, such as *Nurse Supervisor*, *Province*, *Country* and *Sub-Purpose* can be authorized together with the upper layer tables.

- As mentioned before, the inheritance of authorization on Purpose dimension is reversed from the other two dimensions.

- One record could only handle one simple path of each dimension. Authorizations on complex hierarchies require more records. For example, if the patient authorizes Doctor and Nurse to access his/her private data in one disclosure set, two tuples in the fact table's column *"Authorized User-Table"* would be *Doctor* and *Nurse*. Moreover, we can slightly modify this schema to satisfy all

the disclosure options in complex hierarchies.

Chapter 5

IMPLEMENTATION

In this chapter, implementation issues are discussed. Firstly, I introduce the tools and environments selected to implement the e-Health system and the privacy protection components. Secondly, the schemata implementation details are illustrated with experiments on redesigned and existing schemata. Thirdly, I describe implementation details for preferences specification, which is an essential part to preserve privacy in e-Health system. Finally, a scenario for private data access is introduced in section 5.4.

5.1 Implementation Environment

5.1.1 SOA with BEA Weblogic 8.1.2

Loosely coupled software services are more and more popular in business process handling nowadays. Web services bring us easier software integration using XML, we

thus determined to use web service of Service Oriented Architecture (SOA) for our e-Health system implementation. Meanwhile, to provide a friendly interface for users (e.g. patients and doctors) and make it easier to maintain and upgrade the whole system, a web-based platform has been selected, while web pages (JSP or Html .etc) represent the user interface for clients' operations.

There are several popular deploying platforms for web service, such as Apache, the .Net framework, and J2EE. They can implement web service on the same platform for running the web applications. There is a test for the two leading commercial J2EE servers, BEA WebLogic and IBM's WebSphere, plus a solid also-ran, Sybase's EAServer, and the most popular open source J2EE server JBoss. The test [14] evaluates the related management capabilities as well as the support for core web service standards, SOAP, XML-RPC (Remote Procedure Call), WSDL and UDDI, flexible configuration and a set of features such as JMX, JNDI, JMS and JTA that would be expected in any enterprise-class Java platform.

We make use of a powerful and robust web platform, WebLogic Workshop 8.1.2 provided by BEA, to complete the system implementation.

Firstly, all the services are designed in the web service file (*.jws). Database control with connection pooling has been used to access EPR data for different methods with different purposes and different recipients. The SQL statements of data usage for each method are also composed within the database control interface.

For user interfaces, we use jsp files controlled by Java Page Flow [17] which is a feature set built upon a Struts-based Web application programming model. Java Page

Table 11: Web Service Deployment Platforms

Product	Cost	Platforms	Bottom Line
JBoss 3.2.1/ Apache Tomcat 5.0	Free	Java 1.3 (or later) JVMs	1. Excellent support for importing and exporting existing web services 2. Good standard support 3. Require more expertise and initial configuration 4. Lack many of simpler enterprise class configuration and management tools
Sybase EAServer 4.2	From $7500 to $2800 per CPU	AIX, HP-UX, Linux, Windows 2000	1. Good integration between app server and tools, a slick and efficient interface. 2. it's slightly hampered by a heavy reliance on server restarts and a help system that could use a little work
IBM WebSphere Application Server 5.0	From $8000 to $25000 per CPU	AIX, Red Hat Linux, Solaris, Windows NT/2000	1. Allows granular control of the app server and web services and does so with almost equal skill from its GUI and CLI; 2. Confusing installation process that can add up to more than 1GB for even a basic installation
BEA WebLogic Sever 8.1	Starts at $10000 per CPU includes WebLogic Workshop	Linux, Unix, Windows	1. Good integration between tools and server, easy setup, solid web server standards support 2. Excellent IDE 3. Importing and manipulating existing web services would be easier without some additional steps

Flow features include runtime support for the Web application programming model and tools that enable developers to quickly and easily build applications based upon the model and it's based on Model-View-Controller (MVC) architectural pattern. The views (jsp files) and controller (jpf file) are generated by the web service file, recomposed and extended to entire functions programming. The example of our EPR services will be illustrated in later part of this chapter.

5.1.2 Oracle 9i Database Support

Due to the stringent privacy and security requirement for data in various medical systems, especially the data processed and stored in EPR systems, the database product selection is chiefly based on the security features of various commercial database vendors.

We survey the products according to several security features: Authentication, Access Control, Encryption, Auditing, Availability .etc. The candidate commercial database products are Oracle 9i, IBM DB2, Sybase ASE15, Microsoft SQL Server 2005 and an open source database system: MySQL.

Moreover, as we have presented in Section 2.1, virtual private database (VPD) is the aggregation of server-enforced, fine-grained access control, together with a secure application context in Oracle database system. By dynamically appending SQL statements with a predicate, VPD limits access to data at record level and ties the security policy to the table or view itself.

Apparently, we need to limit the data access at the hospital level, because some

Table 12: Main Security Features of Database Products

Product	Authentication	Access Control	Encryption	Auditing
Oracle 9i	Password -based Host-based	FGAC by VPD	TDE, OAS and SSL	Triggers Fine-grained Auditing
MS SQL Server/ 2005	Connection String Kerberos .etc	Execution Context	Digital Certificates SSL	Triggers SQL Trace
IBM DB2	Kerberos .etc Host-based	ID based, Role based, .etc	Column Level , Value Level	DB2 Audit Trace
Sybase ASE15	Password, Kerberos, LDAP	DAC Policy based	SSL, Kerberos	Audit Trail
MySQL	Password		Hashing	

hospitals share the same EPR schema in the same database. Each EPR service belongs to one specific hospital, and the data usage for that service should be limited to the data of that hospital, except that valid authorizations have been made for special purposes, the data can be accessed across hospitals.

Assume that the table EPR is like this: EPR (EPR_ID Number (10), Patient_ID Number (6), Diagnosis varchar2 (30), Hospital_ID Number (3)). Even though administrator could access all the data without any restrictions, if we want to allow applications to access only data in the hospital which it belongs to, VPD can satisfy this demand.

Consequently, when the EPR services use the data, the application connects database with the value of Hospital_ID and execute statement: Select * from EPR;

Here, only the rows where $Hospital\_ID = \{Hospital\}$ are visible for the application.

Moreover, we can also limit data access at the USER level using USER_IDs (either

Patient_ID or Doctor_ID) or the tuples level using Authorization_IDs according to the proposed fine-grained authorization and access control mechanism. Since many kinds of privacy metadata have been designed in the database system, we can limit the access based on these systems and implement the access control not only at application level but also at database level. If we take advantage of these features of Oracle 9i to keep the authorization and access control implementation at database level, the privacy protection can be enhanced.

5.2 Hippocratic Databases Schemata Implementation

In the privacy protection subsystem of our e-Health system, the redesigned schemata based on multi-dimensional HDB, the existing schemata for multi-dimensional HDB and the proposed snowflake metadata schema for hierarchical authorizations are deployed in backend databases. First of all, some issues around schemata implementation will be described in this section. Consequently, some experiments on our redesigned schema and existing HDB schema are illustrated.

5.2.1 Oracle Schemata Implementation

Three schemata: original HDB for multi-dimensional schema, redesigned multi-dimensional HDB and snowflake privacy-authorization schema have been implemented as the EPR schema in a specific user account of the Oracle system.

All the required tables, foreign keys, triggers, sequences, and procedures in the schemata implementation constitute the schemata in our design.

To allow users to specify privacy authorizations, we implement an APPEL preference selector through Web interfaces based on BEA WebLogic 8.1.2. We generate authorizations from the selected options in the Web interfaces using a Java application. The application then stores the generated authorizations in a backend Oracle 9i database. Oracle Enterprise Edition supports fine-grained access control (FGAC) through the VPD feature, which is the aggregation of server-enforced FGAC with a secure application context. Our fine-grained authorizations are implemented using this feature by defining policy functions. We implement three schemata as described in Chapter 3 for fine-grained authorizations based on a flat relational model, for multi-dimensional model without hierarchies, and for hierarchical authorizations, respectively. In practice, however, an application may choose to implement only one of those schemata based on its specific needs.

5.2.2 Experiments

To evaluate the previous designs, we populate our database with the adult dataset from the UCI Machine Learning Repository [34] (which has been used in many related work on data privacy). There are 32561 patients in the dataset. We made following assumptions. Each patient owns three records in our EPR table, and each record is associated with four purposes: *"Treatment"*, *"Research"*, *"Statistics"*, and *"Prescription"*. Table 13 shows that our redesigned fine-grained authorization schema

Table 13: Storage Requirements

#	Table	Attributes Number	Tuples Number	Storage (Bytes)
1	*Patient* in Table 4 or 5	15	32,561	5,242,880
2	*EPR* in Table 4	11	390,732	75,497,472
3	*EPR* in Table 5	9	97,683	16,777,216
4	*Purpose-Authorization* in Table 5	3	390,732	24,117,248
	Summation of #1 and #2		80,740,352 Bytes	
	Summation of #1, #3, and #4		46,137,344 Bytes	

requires significantly less total storage than that of the original HDB design, if the record-level fine-grained authorization is to be enforced. Clearly, with more purposes or attributes that hold more storage consuming data (for example, multimedia data), the difference would be more significant.

5.3 Preferences Negotiation Implementation

Patients specify their preferences in the frontend of our e-Health system. We implement the preferences negotiation using a web interface, and integrate it into the front end of our web service based portal system. There are three important components in the preference negotiation process, which are generating disclosure set, conflict resolution and APPEL preferences. The implementation issues of these three components will be introduced in this section.

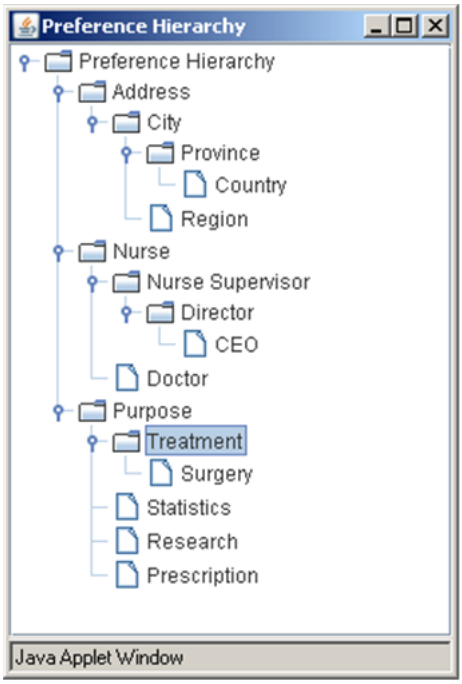

Figure 13: The Hierarchy Used in Implementation

5.3.1 Generating Disclosure Set

In the preference negotiation web interface, there is a hierarchy consisting of the nodes of three dimensions, as illustrated in Chapter 4. Patients are able to make preferences with hierarchical authorizations. A simple example for hierarchies used for disclosure inheritance is shown in Figure 13.

Three dimensions in the implementation are:

- Private Data: Address, City, Province, Country, Region
- Data Recipient: Nurse, Nurse Supervisor, Director, CEO, (Doctor)
- Purpose: Treatment, Surgery, Statistics, Research, Prescription

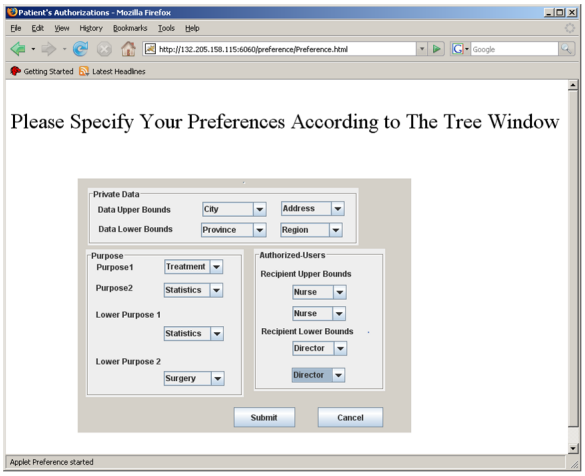

Figure 14: Specifying Preferences in Web Interface

The patient specify the preferences using a web interface shown in Figure 14, in which the patient selects the upper bounds of private data, data recipients and purposes, and their lower bounds. Consequently, a disclosure set based on the selected bounds of those three hierarchies are generated, and the disclosure inheritance are triggered with the generated disclosure set. Finally, a group of authorizations will be inserted into the Table *Privacy-Authorization* (if we still use the original HDB metadata design), or in our redesigned snowflake metadata (the fact table *Privacy-Authorization*).

From the selected bounds shown in Figure 14, there are $4 \times 3 \times 3$ authorizations generated by a single disclosure, and all of the 36 tuples have been inserted into the backend database simultaneously. The differences (before specifying preferences and after specifying preferences) for the rows of the *"Privacy-Authorization Table"* in the database schema is shown in Figure 15 and 16.

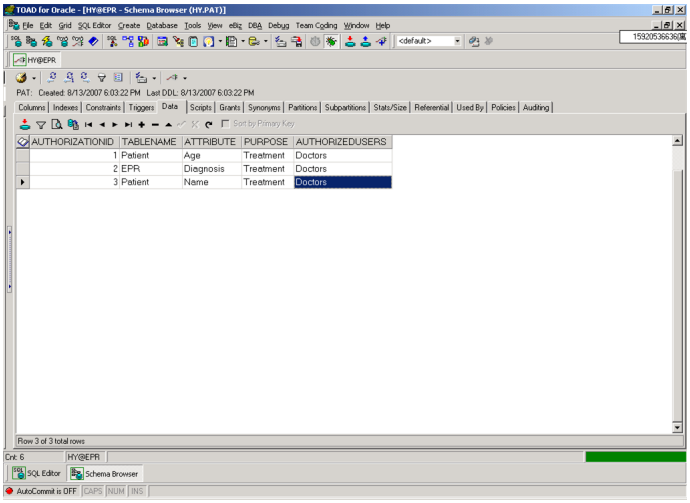

Figure 15: Privacy-Authorization Table before Specifying Preferences

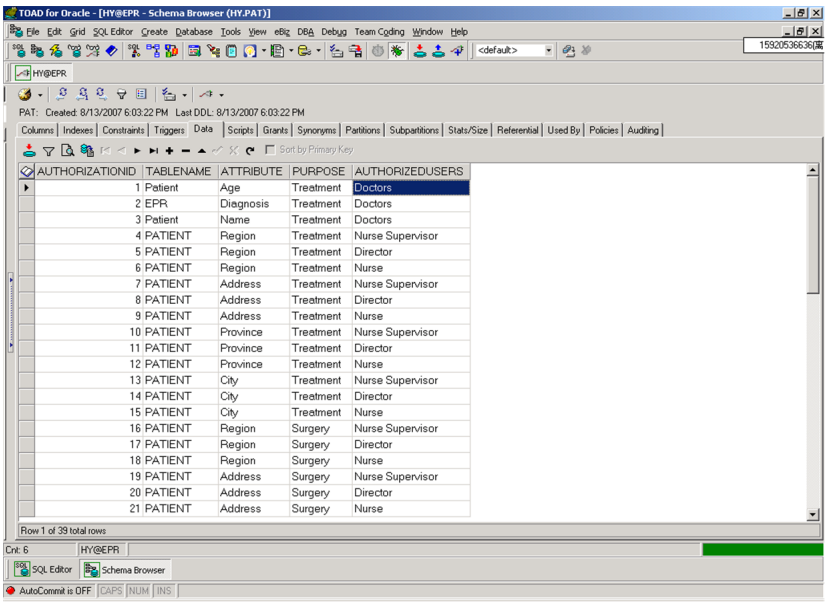

Figure 16: Privacy-Authorization Table after Specifying Preferences

5.3.2 Meta-Policy for Conflict Resolution

Conflict resolution for preferences has been illustrated in Chapter 4. In this subsection, the implementation details will be discussed. There are three meta-policies defined for conflict resolution, which are *"Latest-Take-Precedence"*, *"Disclosure-Take-Precedence"* and *"Denial-Take-Precedence"*. Here, I will take the *"Latest-Take-Precedence"* as the example to describe the meta-policy implementation.

When a patient makes preferences through the web interface, he/she should select the meta-policy if there are some potential conflicts. After a hierarchical authorization has been made by the patient, the system will generate a disclosure set that represents a group of preferences. Consequently, the conflict inspection is triggered.

Conflict inspection is based on the existing preferences, either a single preference or a group of them represented in the snowflake schema, and the new preferences made by the same patient. For example, if the patient disclosed his/her data: *Address*, *City* and *Province*, and the new disclosure set includes *City*, *Country* but not *Province*. During the conflict inspection, a message is displayed: *"Conflict Found!!!"*. The system executes the conflict resolution with the pre-selected meta-policy (such as *"Latest-Take-Precedence"*). Finally, the new disclosure set will be generated. In the above example, the disclosure on *Province* for all the purposes and data recipients will be removed and *Country* will be inserted into the database with corresponding purposes and data recipients. The detailed procedures of the preference negotiation, especially for conflict resolution is shown in Figure 17.

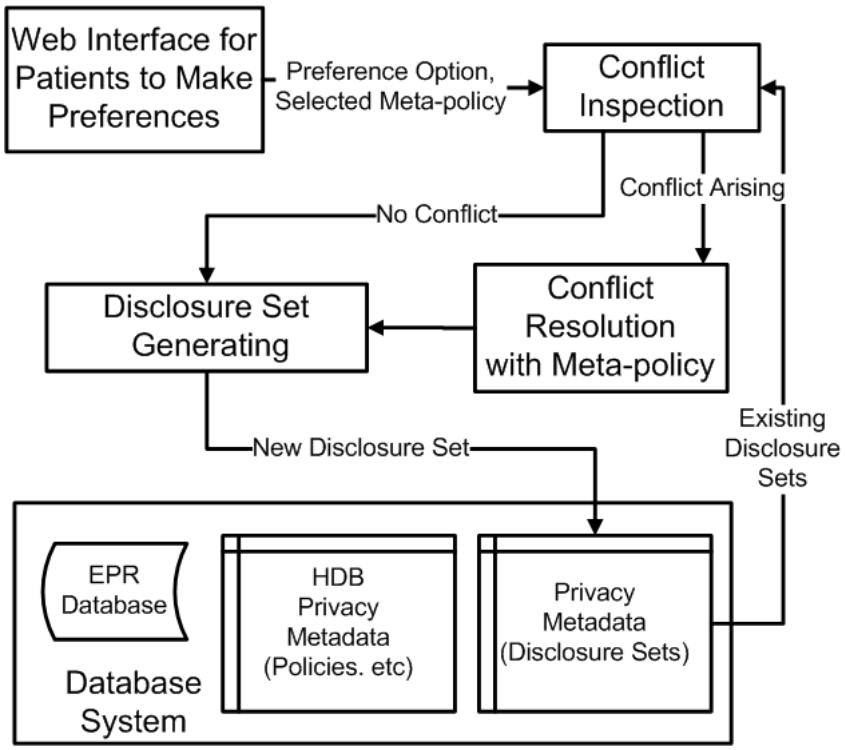

Figure 17: Detailed Preference Negotiation Architecture

5.3.3 APPEL Integration

In previous sections, we presented the server-centric architecture [3] [13]. In this subsection, the tools and resources for building P3P into the service websites will be illustrated.

Referring to the P3P implementation from W3C [11], there are some policies creating tools such as P3PEidt, P3PBuilder, .etc. P3PEdit, is a commercial P3P policy generator whose easy-to-use Web based Wizard could quickly generate P3P policies that satisfy Internet Explore's new privacy requirements. P3PEdit generates: XML format P3P policies, P3P Compact Policies, HTML format Privacy Statements. It includes instructions and examples, technical support, staff review of the P3P implementation, and P3P Policy updates. P3PBuilder is a P3P policy generator that creates privacy policies to the W3C specification, and it's also web based, easy to use.

For creating APPEL preferences, a Java-based editor: JRC APPEL Preference [10] can assist developer to create preferences. The APPEL RULE can be created either by selecting a set of the predefined rules, or by using an advanced mode that provides options to create new rules.

JRC P3P Proxy [10] is an intermediary agent which handles the access control for remote web servers based upon the existing privacy preferences. It will be used for the conformance checking between policies and preferences.

All these three tools with different goals will be combined for the service development and data processing. They performs efficiently in those two architectures, preference negotiation and limited disclosure, which are based on the server-centric

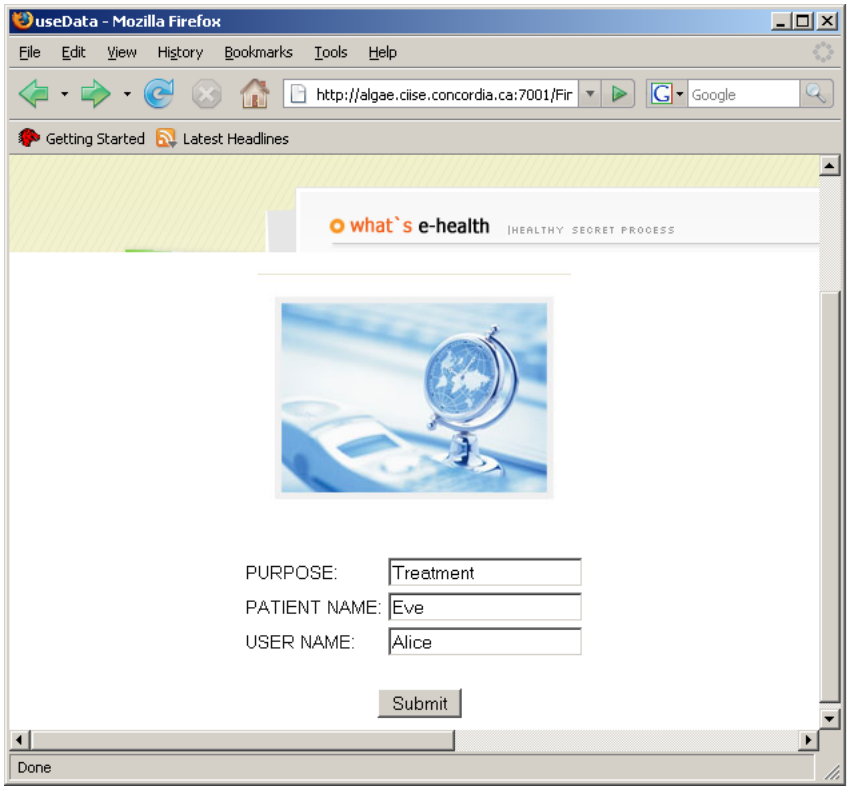

Figure 18: Web Interface for EPR Service to Use Private Data

architecture of P3P.

5.4 A Scenario for Electronic Patient Record Data Access

This section presents a data usage scenario to introduce the privacy protection. The interface is used by various data recipients, such as doctors, nurse. etc.

From the scenario, a doctor (Alice) makes use of the web application, a specific EPR service to read the records of a patient, Eve. She will be authenticated

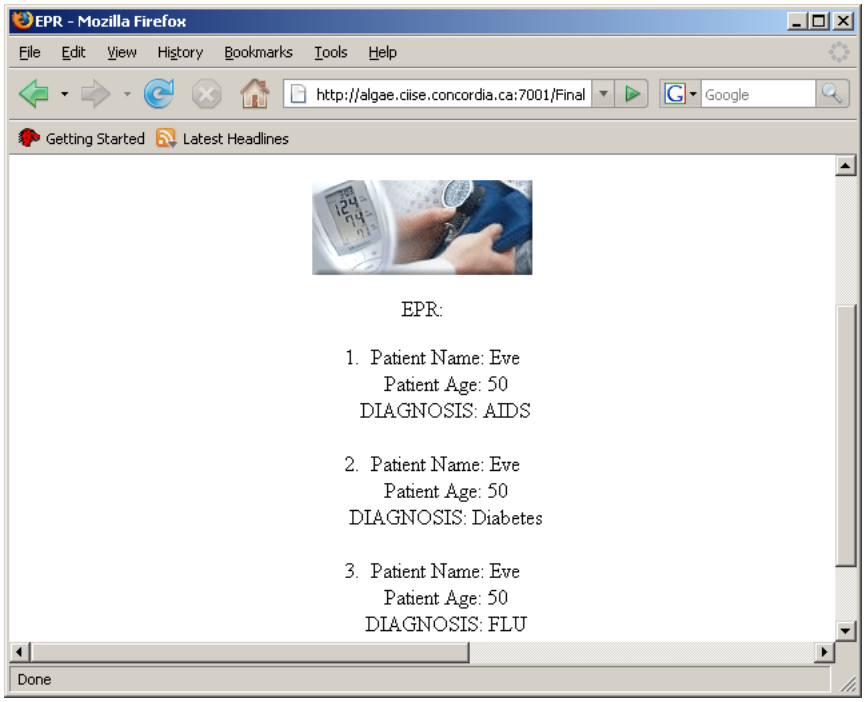

Figure 19: Electronic Patient Records Data Access Based on Authorization

firstly with her credential and purposes, and only the authorized data for her and the matched purposes can be accessed by Alice (a simple example for the web interface is shown in Figure 18). If Eve authorizes doctors to access some of his private data (e.g. Patient Name, Age, and Diagnosis), Alice is able to access those data as a doctor. The results of the queries are presented in Figure 19. Note that all the records for the same patient (Eve) are created at different time and they are totally different. If Eve hasn't authorized doctors to access the requested data, Alice will be refused to view Eve's records using that query. However, if there's no matched records, but Alice is still authorized, an exception will occur that no records are found.

In addition, for the fine-grained authorization in this scenario, only the authorized

tuples in our multi-dimensional EPR schema and the authorized attributes (private data) for Alice with matched purposes can be used by her.

A serial of EPR services including *"Accessing Private Data by Doctor"*, *"Access Private Data by Patient"*, *"Create EPR by Doctor"* are integrated in the e-Health architecture for single sign-on (SSO) with different roles in the system.

Chapter 6

CONCLUSION

This thesis addressed the issue of preserving patients' privacy in e-Health systems. It proposed to allow patients to specify their privacy preferences in APPEL, which will be recorded as authorization policies in the backend HDB. Doctors are then authorized against such policies when their applications request for privacy data. It identified the lack of fine-grained authorizations as a limitation of the original HDB design and provided a solution based on a modified schema. It extended the HDB design to support the multi-dimensional model so it can be used to preserve privacy in analytical data applications. Moreover, it proposed to simplify the task of specifying privacy preferences by exploiting hierarchies inherent to various dimensions of a preference. It also described several options for users to specify a preference based on the hierarchies. It studied how to generate the effective disclosure set in each case and how to resolve potential conflicts that may arise between preferences with pre-defined meta-policies. It showed how hierarchies can be leveraged to simplify the

representation of authorizations, and experimental results justify the design. It is my belief that the proposed solution can be integrated into existing e-Health systems to provide patients with better privacy protection.

Bibliography

[1] United States Public Law 104-191. Health insurance portability and accountability act. 1996.

[2] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kiernan, R. Rantzau, and R. Srikant. Auditing compliance with a hippocratic database. *Proc. of the 30th In'tl Conf. on Very Large Databases (VLDB)*, 2004.

[3] R. Agrawal, J.Kiernan, R. Srikant, and Y. Xu. Implementing p3p using database technology. *Proc. of the 19th International Conference on Data Engineering (ICDE)*, 2004.

[4] R. Agrawal and C. Johnson. Securing electronic health records without impeding the flow of information. *International Journal of Medical Informatics*, 2007.

[5] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. *Proc. of the 28th Int'l Conference on Very Large Databases (VLDB)*, 2002.

[6] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Server-centric p3p. *Proc. of the W3C Workshop on the Future of P3P*, 2002.

[7] R. Agrawal and R. Srikant. Privacy-preserving data mining. *Proc. of the ACM SIGMOD Conference on Management of Data*, 2000.

[8] R. J. Anderson. A security policy model for clinical information systems. *Proc. of the IEEE Symposium on Security and Privacy*, 1996.

[9] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. *Proc. of the 21st International Conference on Data Engineering (ICDE'05)*, 2005.

[10] JRC P3P Resource Centre. Appel implementation. *http://p3p.jrc.it/.*

[11] World Wide Web Consortium. P3p implementations from w3c. *http://www.w3.org /P3P/implementations.*

[12] L. Cranor, M. Langheinrich, and M. Marchiori. A p3p preference exchange language 1.0 (appel 1.0). 2002.

[13] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler Marshall, , and J. Reagle. The platform for privacy preferences 1.0 (p3p 1.0) specification. 2002.

[14] Products for Java Web Service Evaluation. *http://www.infoworld.com/article/...*

[15] Y. Hong, S. Lu, Q. Liu, L. Wang, and R. Dssouli. A hierarchical approach to the specification of privacy preferences. *in Proceedings of the 2007 International Conference on Innovations in Information Technology*, 2007.

[16] Y. Hong, S. Lu, Q. Liu, L. Wang, and R. Dssouli. Preserving privacy in e-health systems using hippocratic databases. *in Proceedings of the 32nd Annual International Computer Software and Applications Conference*, 2008.

[17] BEA Inc. Java page flow. *http://sybooks.sybase.com/onlinebooks.*

[18] IBM Inc. Ibm db2 administration guide.

[19] IBM Inc. Ibm hippocratic database active enforcement user guide version 1.0 -including technical appendices.

[20] Microsoft Inc. Implementing row- and cell-level security in classified databases using sql server 2005. *http://www.microsoft.com/technet/prodtechnol/sql/2005/multisec.mspx.*

[21] Sybase Inc. Sybase adaptive server enterprise 12.5, system administration guide, row level access control. *http://sybooks.sybase.com/onlinebooks.*

[22] K. Irwin and T. Yu. Determining user privacy preferences by asking the right questions: An automated approach. *Proc. of the 2005 ACM workshop on Privacy in the electronic society*, 2005.

[23] B. Jiang, D. Zhang, C. Yang, G. Yue, and F. Yu. A privacy policy of p3p based on relational database. *Proc. of the 1st International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06)*, 2006.

[24] G. Kaur and N. Gupta. E-health: A new perspective on global health. *Journal of Evolution and Technology*, 15(1), 2006.

[25] R. Kirkgoze, N. Katic, M. Stolba, and A. Min Tjoa. A security concept for olap. *Proc. of the 8th International Workshop on Database and Expert Systems Applications*, 1997.

[26] B. Kolman, R. C. Busby, and S. C. Ross. *Discrete Mathematical Structures.* 2000.

[27] T. Kyte. Fine-grained access control. *Oracle Corporation*, 1999.

[28] J.-Gil Lee, K.-Young Whang, W.-Shin Han, and I.-Yeol Song. Hippocratic xml databases : A model and an access control mechanism. *Journal of Computer Systems Science and Engineering*, 21(6), 2006.

[29] Q. Liu, S. Lu, Y. Hong, L. Wang, and R. Dssouli. Securing tele-health applications in a web-based e-health portal. *in Proceedings of the Third IEEE International Conference on Availability, Reliability and Security*, 2008.

[30] S. Lu, Y. Hong, Q. Liu, L. Wang, and R. Dssouli. Access control in e-health portal systems. *in Proceedings of the 2007 International Conference on Innovations in Information Technology*, 2007.

[31] F. Massacci, J. Mylopoulos, and N. Zannone. Minimal disclosure in hierarchical hippocratic databases with delegation. *Proc. of the 2005 Computer Security-ESORICS*, 3679.

[32] Arup Nanda and Donald K. Burleson. *Oracle privacy security auditing: includes federal law compliance with HIPAA, Sarbanes-Oxley and the Gramm-Leach-Bliley Act GLB.* 2003.

[33] N. Pendse. The olap report - what is olap. *http://www.olapreport.com/fasmi.html.*

[34] UCI Machine Learning Repository. Adult dataset. *http://archive.ics.uci.edu/beta/datasets/Adult.*

[35] P. Samarati and S. de Capitani di Vimercati. Access control: Policies, models, and mechanisms. *Springer Berlin / Heidelberg*, 2171/2001, 2004.

[36] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. *Proc. of the 7th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1998.

[37] Lorrie Faith Cranor Simon Byers and David Kormann. Automated analysis of p3p-enabled web sites. *Proc. of the 5th International Conference on Electronic Commerce (ICEC)*, 2003.

[38] S. Sung, Y. Liu, H. Xiong, and P. Ng. Privacy preservation for data cubes. *Proc. of the 20th International Conference on Data Engineering (ICDE)*, 2004.