HIERARCHICAL HIPPOCRATIC DATABASES WITH MINIMAL DISCLOSURE FOR VIRTUAL ORGANIZATIONS

Fabio Massacci, John Mylopoulos, and Nicola Zannone

April 2006

Technical Report # DIT-06-021

Hierarchical Hippocratic Databases with Minimal Disclosure for Virtual Organizations

Fabio Massacci
University of Trento
massacci@dit.unitn.it

John Mylopoulos
University of Trento
jm@dit.unitn.it

Nicola Zannone
University of Trento
zannone@dit.unitn.it

Abstract

The protection of customer privacy is a fundamental issue in today's corporate marketing strategies. Not surprisingly, many research efforts have proposed new privacy-aware technologies. Among them, Hippocratic databases offer mechanisms for enforcing privacy rules in database systems for inter-organizational business processes (also known as virtual organizations). This paper extends these mechanisms to allow for hierarchical purposes, distributed authorizations and minimal disclosure supporting the business processes of virtual organizations that want to offer their clients a number of ways to fulfill a service. Specifically, we use a goal-oriented approach to analyze privacy policies of the enterprises involved in a business process. Based on the purpose hierarchy derived through a goal refinement process, we provide algorithms for determining the minimum set of authorizations needed to achieve a service. This allows us to automatically derive access control policies for an inter-organizational business process from the collection of privacy policies associated with different participating enterprises. By using effective on-line algorithms, the derivation of such minimal information can also be done on-the-fly by the customer wishing to access a service.

Keywords Privacy Protection, Minimal Disclosure, Private Data Management, Information Security, Access Control, Virtual Organizations, Delegation

1 Introduction

In the last few years data and privacy protection have become critical issues in the development of information systems. This reflects the growing attention of customers to their personal information and the increasing number of laws, policies, and regulations that are intended to safeguard it. The US Privacy Act of 1974 and the EU Directives on Privacy in 1995 (and later regulations) define privacy as the right of data subjects to determine how their personal data are used. Several proposals [5, 7, 10, 17, 20, 28] introduce the concept of *purpose* in order to capture this definition where purpose represents the intended usage of information.

Together with the notion of purpose, current privacy legislations also define the privacy principles that an information system has to meet in order to guarantee customer privacy. At the basis of the exchange between enterprises and customers, there is the principle of transparency: enterprises should disclose to customers which data are collected and for what purpose. Another important principle is the notion of minimal disclosure: enterprises should maintain only such information about an individual as is necessary to fulfill the purpose for which it was collected.

The transparency principle should aid customers to verify whether or not enterprises implement the minimal disclosure principle correctly. For transparency, enterprises should declare in their privacy policies the purpose for which data are collected, who can receive them, the length of time the data can be retained, and the authorized users who can access them. Looking at such policies customers would be able to understand how their personal data will be used and, in case they agree, disclose them. Obviously, if an enterprise requires more data than some customers feel needed for the desired service, they can get their services elsewhere.

In general, it is up to customers to decide on a strategy of how to get a service fulfilled on the basis of their personal feeling of trust for any one enterprise. However, this decision can be very difficult when an enterprise provides many ways to achieve a service. It gets worse when we do not have a single enterprise that delivers the service by itself, but rather a set of collaborating organizations participating in a single business process (*virtual organization* [16]).

Virtual organizations offer services that can be dynamically customized in at least two different ways:

1. different service components are dynamically chosen for fulfilling the same high level goal, but possibly using different data;

2. different partners (sub-contractors) are chosen for fulfilling the same goal, but possibly with different service level agreements or trust levels.

The choice of service customization has significant impact on the privacy of individual customers. Different customizations may require different data for which privacy considerations vary; there might be trusted and untrusted partners offering the same service. Consider an example. Mississippi, an on-line book company, notifies the status of the order to its customers through email or Short Message Service (SMS). Customers are not interest in a generic process; rather, they want the process that best protects their privacy based on their preferences. Depending on the preference of the customer, Mississippi needs either an email address or a mobile phone number. Alice, a professor plagued by spam, may treasure her email address and give away her business mobile phone number. Bob, a doctor whose mobile phone is always ringing, may have the opposite preference. The partners chosen by Mississippi might also be trusted differently by its potential customers. Bob may be unwilling to give his email to a professional delivery company, because last time he did, he got dozens of emails notifying him unwanted personalized services.

We are interested in solutions that support customers and companies alike, so that companies can publish comprehensive privacy policies involving hierarchies of purposes, possibly spanning multiple partners. Moreover, we want our solution to allow customers to personalize services they want on the basis of their own privacy sensitivities and their trust of partners who might contribute to the delivery of a requested service.

1.1 Contribution of the paper

This paper proposes enhancements to Hippocratic database systems [2] in order to deal with inter-organizational business processes managed by virtual coalitions. In particular, we present a flexible framework for automatically deriving the minimum set of authorizations needed to achieve a service from enterprise privacy policies when a host of partners participating in the business process provides different ways to achieve the same service.

Our work is grounded on modeling and analysis of purposes for Hippocratic databases, using goal-oriented approaches [6]. Our models organize purposes into AND/OR tree hierarchies. The framework allows customers to express their preferences in the form of privacy penalties associated with each personal data item and each partner of the business process. Thus, the process for fulfilling a purpose can be customized at run-time and guarantees maximal privacy protection because it was selected with criterion of the smallest privacy penalty.

Similar approaches have already been used in Requirements Engineering in order to represent software requirements and reason about their fulfillment by adopting different solutions. In this context, Sebastiani et al. [25] presented a formal framework for reasoning with goal models in a qualitative and quantitative way. In particular, they proposed algorithms to find the minimum cost assignment of labels to leaf goals which satisfies root goals. However, their solutions are not adequate for our purposes since they are designed for off-line analysis by the system designer and do not allow customers to set their preferences on-line. Accordingly, we propose to model hierarchies using hypergraphs [3, 4] since this data structure is suitable for studying reachability and minimum weight traversal problems and efficient algorithms already exist. We use these data structures to represent the privacy policies of each partner of a business process as a weighted directed acyclic graph (weighted DAG). Then, we merge such DAGs in order to build a DAG representing the privacy policies governing the entire business process where edges across DAGs are seen as delegations of customer information among the partners of the business process. Weights (or privacy penalties) are specified by customers and represent the cost of disclosed information.

Based on this data structure, we provide algorithms for finding the minimal decomposition path that represents the process with the smallest privacy penalty. To support the dynamic customization of the process, we also give algorithms for efficiently updating it when customers change the cost of data items or choose among the alternatives that an enterprise offers for achieving a required service. Then, this decomposition path is used to build the minimal privacy authorization table that represents the minimum set of authorizations necessary to achieve a service according to customer preferences.

1.2 Outline

The remainder of the paper is structured as follows. Section 2 introduces a scenario that is used as a running example throughout the paper. Section 3 presents a brief description of Hippocratic database systems. Section 4 introduces purpose DAGs in order to represent purpose hierarchies, while Section 5 discusses how to build a purpose DAG from a Hippocratic database system. Section 6 shows how to derive the minimal privacy authorization table and Section 7 describes the data structures used to represent purpose DAGs. Sections 8 and 9 present respectively algorithms for finding and updating the minimum cost path, and a comparison between the minimal privacy authorization table and the privacy authorization tables derived using Agrawal's approach. Finally, Section 10 discusses related work, and Section 11 concludes the paper with some directions for future work.

2 A Running Example

This section presents a scenario used throughout the paper. The scenario is a revised version of the case study proposed by Agrawal et al. [2].

Mississippi is an on-line book store and its offerings include an on-line catalogue to its customers where they can search for the items they wish to buy. Customers can add selected items to their shopping basket. Once customers have completed their selection, they need to create a new account or login to their existing account in order to enter their delivery and payment details, since Mississippi needs to obtain

Table 1: Database Schema

table	attribute
customer	purpose, customer-id, name, address, email, mobile-number, credit-card-info
order	purpose, customer-id, transaction, book-info, status

Table 2: Privacy Metadata Schema

table	attributes
privacy-policies	purpose, table, { attribute }, { external-recipients }, retention
privacy-authorizations	purpose, table, { attribute }, { authorized-users }

certain personal information from customers in order to perform purchase transactions. This information includes name, shipping address, and credit card number.

Mississippi views a purchase, its ultimate purpose, as a three-step process: credit assessment, delivery, and notification. Mississippi delivers books through either a delivery company or the post office. Notification is used to communicate the status of the order to the customer, and it is accomplished by Mississippi either through email or Short Message Service (SMS). Depending on the method, Mississippi needs either email or mobile phone number information about each customer.

For shipping books, Mississippi relies on Worldwide Express (WWEx). WWEx is a delivery company that offers a global network of specialized services – transportation, international trade support and supply chain services. WWEx needs to obtain customer information to deliver books. This information includes customer name and shipping address. WWEx is usually not able to complete a delivery on its own, but depends on local delivery companies for door-to-door delivery. To this end, WWEx delegates customer information to them. In the remainder of the paper, we call $\text{LDC}_1, \ldots, \text{LDC}_n$ the local delivery companies responsible to deliver books in the zone where the customer lives.

Furthermore, Mississippi relies on the Credit Card Company (CCC) for credit assessment. CCC needs some customer information as well to provide its service. This information includes customer name and credit card number, and the transaction between Mississippi and the customer. For making credit decisions, CCC needs a credit rating[1] for which it depends on the Credit Rating Company (CRC). CRC uses statistics to summarize past experience so that predictive analysis can be used to generate a rating for the customer. Based on the rating, CCC decides to accept or not the customer transaction and then communicates its decision to Mississippi.

3 An Overview of Hippocratic Databases

Based on current US and EU privacy legislation, Agrawal et al. identified the privacy principles an information system should meet in order to enforce privacy and data protection: purpose specification, consent, limited collection, limited use, limited disclosure, and limited retention. Based on such principles, the authors propose Hippocratic databases [1, 2] which were designed to support them. In the remainder of

[1]Credit rating is a method for interpreting the content of a credit report. Credit rate represents the probability of a fail-to-pay event happening in the future, but does not indicate that such an event will happen.

Table 3: Mississippi's Privacy-Policies Table

purpose	**table**	**attributes**	**external-recipients**	**retention**
purchase	customer	{name, address, email, mobile-number, credit-card-info}	*empty*	1 month
purchase	order	{transaction, book-info, status}	*empty*	1 month
delivery	customer	{name, address}	*empty*	1 month
direct delivery	customer	{name, address}	{ delivery-company }	1 month
delivery by post	customer	{name, address}	{ post-office }	1 month
credit assessment	customer	{name, credit-card-info}	{ credit-card-company }	1 month
credit assessment	order	{transaction}	{ credit-card-company }	1 month
notification	customer	{name, email, mobile-number}	*empty*	1 month
notification	order	{book-info, status}	*empty*	1 month
notification by email	customer	{name, email}	*empty*	1 month
notification by email	order	{book-info, status}	*empty*	1 month
notification by SMS	customer	{name, mobile-number}	*empty*	1 month
notification by SMS	order	{book-info, status}	*empty*	1 month

Table 4: CCC's Privacy-Policies Table

purpose	**table**	**attributes**	**external-recipients**	**retention**
credit assessment	customer	{name, credit-card-info}	*empty*	1 month
credit assessment	order	{transaction}	*empty*	1 month
credit scoring	customer	{credit-card-info}	{ credit-reference-agency }	1 month
credit resolution	customer	{name, credit-card-info}	*empty*	1 month
credit resolution	order	{transaction}	*empty*	1 month

the section, we give an overview on Hippocratic databases looking at how this system enforces privacy principles.

Hippocratic databases use *purpose* as a central concept. Purpose is stored in the database as a "special" attribute occurring in every table of the database. This attribute specifies the purpose (reason/goal) for which a piece of information can be used.

Example 1 *Table 1 shows the schema of tables* customer *and* order *that store the information collected by Mississippi. In particular, table* customer *stores personal information about customers, and table* order *stores information about the transaction between Mississippi and its customers.*

In the sequel we will use the term *user* to denote an employee of a company, that is, a user of the database (rather than a customer of the company). In the terminology of privacy legislation, a user is called *data processor*, while the customer is called *data subject*.

Then, for each purpose and for each data item stored in the database, the following fields are defined:

- *external-recipients*, i.e. the actors to whom the data item is disclosed;
- *retention-period*, i.e. the period during which the data item should be maintained in the database;
- *authorized-users*, i.e. the users entitled to access the data.

Table 5: WWEx's Privacy-Policies Table

purpose	**table**	**attributes**	**external-recipients**	**retention**
direct delivery	customer	{name, address}	*empty*	1 month
door-to-door delivery	customer	{name, address}	{ local-delivery-company }	1 month

Table 6: Mississippi's Privacy-Authorization Table

purpose	**table**	**attributes**	**authorized-users**
purchase	customer	{name, address, email, mobile-number, credit-card-info}	{ Mississippi }
purchase	order	{transaction, book-info, status}	{ Mississippi }
delivery	customer	{name, address}	{ Mississippi }
direct delivery	customer	{name, address}	{ Mississippi, WWEx }
delivery by post	customer	{name, address}	{ Mississippi, Post Office }
credit assessment	customer	{name, credit-card-info}	{ Mississippi, CCC }
credit assessment	order	{transaction}	{ Mississippi, CCC }
notification	customer	{name, email, mobile-number}	{ Mississippi }
notification	order	{book-info, status}	{ Mississippi }
notification by email	customer	{name, email}	{ Mississippi }
notification by email	order	{book-info, status}	{ Mississippi }
notification by SMS	customer	{name, mobile-number}	{ Mississippi }
notification by SMS	order	{book-info, status}	{ Mississippi }

Purpose, external recipients, authorized users, and retention period are stored in the database on the basis of the metadata schema defined in Table 2 [2]. Specifically, the above information is split into separate tables: external-recipients and retention period are in the *privacy-policies* table, while authorized-users are in the *privacy-authorizations* table. The purpose is stored in both. The privacy-policies table contains the privacy policy of an enterprise.

Example 2 *Mississippi's privacy-policies table is shown in Table 3, CCC's privacy-policies table in Table 4, and WWEx's privacy-policies table in Table 5.*

The privacy-authorizations table contains the access control policies enforcing privacy policies. Privacy-authorizations tables are derived from privacy-policies tables by instantiating each external recipient with the corresponding authorized users. Thus, these tables represent what information is actually disclosed.

Example 3 *Tables 6, 7 and 8 show Mississippi, CCC and WWEx's privacy-authorizations tables, respectively. In particular, Table 6 shows that Mississippi can access both email address and mobile phone number for notifying the status of an order, and that WWEx and Post Office can access customer data for direct delivery and delivery by post, respectively. Notice that these authorizations match exactly the policies stated in the corresponding privacy-policies table. Moreover, looking at Table 8, we can see that all local delivery companies are authorized to access customer personal data.*

The consent principle is enforced by Hippocratic systems through the *Privacy Constraint Validator*. This module verifies whether customer preferences match the privacy policies of the enterprise before the

Table 7: CCC's Privacy-Authorization Table

purpose	table	attributes	authorized-users
credit assessment	customer	{name, credit-card-info}	{ CCC }
credit assessment	order	{transaction}	{ CCC }
credit scoring	customer	{credit-card-info}	{ CCC, CRC }
credit resolution	customer	{name, credit-card-info}	{ CCC }
credit resolution	order	{transaction}	{ CCC }

Table 8: WWEx's Privacy-Authorization Table

purpose	table	attributes	authorized-users
direct delivery	customer	{name, address}	{ WWEx }
door-to-door delivery	customer	{name, address}	{ WWEx, $\mathrm{LDC}_1, \ldots, \mathrm{LDC}_n$ }

customer discloses his information. If it is the case, the privacy authorization table is created; otherwise, the information about the customer is not collected.

When a user submits a query to the database, the system not only verifies that the user is authorized to access the required data items, but answers only queries for which the purpose is equal to that for which the information has been collected. Further, Hippocratic databases do not disclose information for purposes other than those for which the owner of the information has previously given consent. Thus, Hippocratic databases implement, respectively, the limited use and disclosure principles. Hippocratic databases enforce the retention principle using the *Data Retention Manager*. Essentially, this module deletes data items when their retention period is expired.

The limited collection principle requires that enterprises collect only the information strictly needed to fulfill the purpose for which data are stored. Hippocratic databases implement the principle through three components: *Access Analysis*, *Granularity Analysis*, and *Minimal Query Generation*. The first module identifies for each purpose which data items never occur in query answers. The second determines the granularity of the required information. Finally, Minimal Query Generation designs queries that disclose the minimum set of information needed for fulfilling a certain purpose. Notice that Access Analysis can only detect those data items that are never used. Therefore, it is sufficient that Mississippi notifies the status of the order to different customers using both methods (SMS and email), so that Access Analysis is not able to detect the redundant information required by privacy policies from single customers. Moreover, Access Analysis works only on data items, and it cannot prevent disclosure of customer information to all local delivery companies.

4 Hierarchy and Delegation of Purposes

The approach proposed by Agrawal is elegant and simple, but does fare well with dynamic situations often encountered in eCommerce-related contexts. First, enterprises generally provide their services in different ways. Then, enterprises might need to decompose a generic purpose into more specific ones since they are not completely able to fulfill it by themselves, and so they may delegate the fulfillment of sub-purposes to third parties. This is the case for a business process where different partners explicitly combine their efforts into one process in order to provide a service to customers. These issues mainly affect the creation of the privacy authorization table since the privacy policy table cannot be directly mapped to it without introducing authorizations unnecessary for fulfilling a service.

As a partial solution Agrawal et al. [2] propose to decompose purposes into multiple sub-purposes and then store them in the database. Based on this, customers can opt in or out of making personal information available. However, using this simple notion of purpose/sub-purpose hierarchy we lose the logical relation between a purpose and its sub-purposes. In particular, this notation does not distinguish whether a sub-purpose is derived through an AND- or OR-decomposition. Consequently, it does not allow reasoning about the fulfillment of the root purpose. For example, a customer might opt out of providing information necessary to fulfill a sub-purpose that, however, is necessary to fulfill the root purpose. Therefore, the enterprise may collect from the customer information for sub-purposes that is altogether insufficient to fulfill the root purpose. Yet, by adopting Agrawal's proposal the enterprise has no automatic mechanisms for detecting such situations.

Following goal analysis approaches [21], we propose to decompose purposes into sub-purposes through an AND/OR refinement. Essentially, AND/OR refinement combines AND- and OR-decompositions of purposes into sub-purposes, modeling a logical purpose structure. Then,

- if purpose p is AND-decomposed into sub-purposes $p_1, \ldots, p_n$, then all of the sub-purposes must be satisfied in order to satisfy p,
- if purpose p is OR-decomposed into sub-purposes $p_1, \ldots, p_n$, then at least one of the sub-purposes must be satisfied in order to satisfy p.

In essence, AND-decomposition is used to define the process for achieving a purpose, while OR-decomposition defines alternatives for achieving a purpose.

Once we have built such a hierarchy, we need a data structure to represent it in order to design algorithms for reasoning about the fulfillment of root purposes. Our choice for this is hypergraphs [3, 4], where AND- and OR-decompositions can be represented as hypergraph edges. In the remainder of the paper, we call the hypergraph representations of purpose hierarchies *purpose directed acyclic graph* (*purpose DAG*). Next, we define it formally.

Definition 1 *A* purpose DAG $\mathcal{P}$ *is a pair* $\langle P, D\rangle$ *where* P *is a set of purposes and* D *is a set of decomposition arcs. Each decomposition arc is an ordered pair* $\langle S, t\rangle$ *from an arbitrary nonempty set* $S \subseteq P$ *(*source set*) to a single node* $t \in P$ *(*target node*).*

If a purpose DAG $\mathcal{P}$ is represented by adjacency lists, the *size* of its description is $|\mathcal{P}| = p + a + d$ where p is the number of purpose nodes, d is the number of decomposition arcs, and $a = \sum_{S \in \mathcal{S}} |S|$ is the *source area*, that is, the sum of cardinalities of all source sets where $\mathcal{S} = \{S | \langle S, t\rangle \in D \text{ for some } t \in P\}$ denotes the *set of source sets*.

Purpose DAGs can be used to represent goal models in goal-oriented Requirements Engineering approaches [6]. For our purposes, they represent the entire set of alternative ways for delivering a service required by customers. Such representations can also be used to model the delegations of tasks and authorizations in the security modeling methodology proposed by Giorgini et al. [15]. Actually, the privacy policies of each partner of a business process can be represented in terms of a purpose DAG. Merging all these DAGs allows us to get a new purpose DAG representing the privacy policies governing the entire business process.

Definition 2 *Let $\mathcal{P} = \langle P, D\rangle$ be a purpose DAG. A purpose DAG $\mathcal{P}' = \langle P', D'\rangle$ such that $P' \subseteq P$ and $D' \subseteq D$ and, for each $\langle S, t\rangle \in D'$, $S \subseteq P'$, is called a* sub-purpose DAG *of $\mathcal{P}$. This is denoted by $\mathcal{P}' \subseteq \mathcal{P}$.*

An enterprise could provide different methods to achieve a service or rely on different partners to achieve the same part of the service. Consequently, different processes can be used to fulfill the required service. To capture this insight, we introduce the notion of *decomposition path*. Roughly speaking, a decomposition path is a particular sub-purpose DAG representing a possible solution through which an enterprise can fulfill a purpose.

Definition 3 *Let $\mathcal{P} = \langle P, D\rangle$ be a purpose DAG, $Z \subseteq P$ be a non-empty subset of purposes, and t be a purpose in P. A* decomposition path *$\mathcal{D}_{Z,t}$ is a set of decomposition arcs $D' \subseteq D$ such that either $t \in Z$ or there exists a decomposition arc $\langle S, t\rangle \in D'$ and there are decomposition paths $\mathcal{D}_{Z,x} \in D'$ for each $x \in S$.*

Since our reference business model is that of virtual organizations, we assume that there will often be more than one way to deliver a service, i.e., different decomposition paths that fulfill the same purpose (or sub-purpose). Yet, they may differ in an important aspect, notably they may require different private data items. Thus, depending on each customer's individual preferences, the same decomposition path might have a significantly different privacy "cost" for different customers. Hence a key issue is to determine the "minimal" decomposition path through a quantitative analysis. To this end, we introduce the notion of weighted purpose DAGs.

Definition 4 *A* weighted purpose DAG *$\mathcal{P} = \langle P, D\rangle$ is one where each decomposition arc $\langle S, t\rangle \in D$ has associated with it a weight $\omega_{\langle S,t\rangle}$.*

Purpose DAGs can have a complex structure, and different cost functions can be used to evaluate the cost of a decomposition path. Depending on the choice, the problem of computing such cost can be polynomially tractable [14] or NP-hard [8, 11, 23]. Specifically, the problem of finding the minimal cost hyperpath in a directed hypergraph has been shown to be NP-hard when the cost of a hyperpath is the sum of the weights of its hyperarcs [3, 4]. In contrast, polynomial time algorithms exist if the cost function is additive [3, 4, 19]. For additive cost functions, the cost of one edge may be counted as many times as it is used. Essentially, additive cost functions work on the unfolded representation of the hypergraph.

The choice of the cost function depends on the application. In our application the choice between two mathematical functions can be transformed into a choice between two philosophical approaches to the disclosure of private data: if we care whether data are disclosed at all, then the mathematical functions that determine the privacy penalty of a decomposition path is the sum of the weights of its arcs. If we also care for the number of times that our private data are used or for the number of business partners that have access to them, then the mathematical cost function that we should use is additive. For our purposes, we choose the second standpoint because we argue that, the more a piece of data is used, the more it is likely that it might be misused. Therefore, additive measures are the ones that capture best one's intuitions on the protection of privacy. In order to implement an additive cost function, we associate the cost of a decomposition path $\mathcal{D}_{Z,t}$ to the node t.

Definition 5 *Let Z be a source set, t be a purpose node, and $\mathcal{D}_{Z,t}$ be a decomposition path from Z to t. The* disclosure penalty *(or* privacy penalty*) to reach t starting from Z, $dp(t)$, is inductively defined as follows:*

- For each supplier privacy policy table, purposes are analyzed through a goal refinement process.
- Once a DAG for each supplier is defined, the DAG representing the privacy policies of the entire business process is built by merging them.
- Each purpose is associated with the data items directly needed to achieve it.
- A privacy penalty is associated with each decomposition arc.

Figure 1: Mapping Hippocratic DB into Purpose DAGs

1. *if* $t \in Z$, *then* $dp(t) = 0$*;*
2. *if path* $\mathcal{D}_{Z,t}$ *has root* $\langle S, t\rangle$ *with subpaths* $\mathcal{D}_{Z,x_1}, \ldots, \mathcal{D}_{Z,x_k}$, *then* $dp(t) = \omega_{\langle S,t\rangle} + \sum_{x_i \in S} dp(x_i)$*.*

5 From Hippocratic DB to Purpose DAGs

We want to determine the process by which a service can be delivered with minimal privacy costs. This section proposes a procedure for building the purpose DAG representing the privacy policies of a business process consisting of many different partners from the Hippocratic database system of each partner.[2] A sketch of the procedure is given in Fig. 1.

Firstly, purposes stored in the privacy policies table of each supplier are analyzed through a goal refinement process based on AND- and OR-decompositions. Graphically, these purpose DAGs are circumscribed by a broken line and labeled with the name of the partner.

These DAGs allow customers to select alternatives proposed by single partners of the business process, but they do not help in determining the minimum cost process for fulfilling the full service. Therefore, once we have a purpose DAG for each supplier, we build the purpose DAG representing the privacy policies of the entire business process by merging them. The merge is done by looking at the external-recipients field stored in every privacy policies table: when the external-recipients field is not empty, we join its purpose with the corresponding purpose (with the same name) occurring in the DAG associated with the partner that is an instance of some external recipient. The purpose node is associated only with the purpose DAG representing the partner. Arcs linking nodes across DAGs are called *delegation arcs*. If there is more than one instance for the same external recipient, we create a number of "copies" of that purpose equal to the number of instances. Every such purpose node is linked to the upper level purpose. Essentially, this process corresponds to an OR-decomposition of a purpose into all the possible alternatives that can be used to achieve it. This solution is also used when there are multiple external suppliers for the same purpose.

This approach supports complex enterprise strategies and, at the same time, allows customers to directly choose a certain supplier whenever the choice is available. Notice that building the purpose DAG representing the privacy policies of the entire business process requires a common ontology among all

[2]We assume knowledge of the complete business process. This is actually closer to reality than one may think: privacy legislations require that every enterprise declares and enforces its own privacy policies and is responsible for the privacy policies of its sub-contractors.

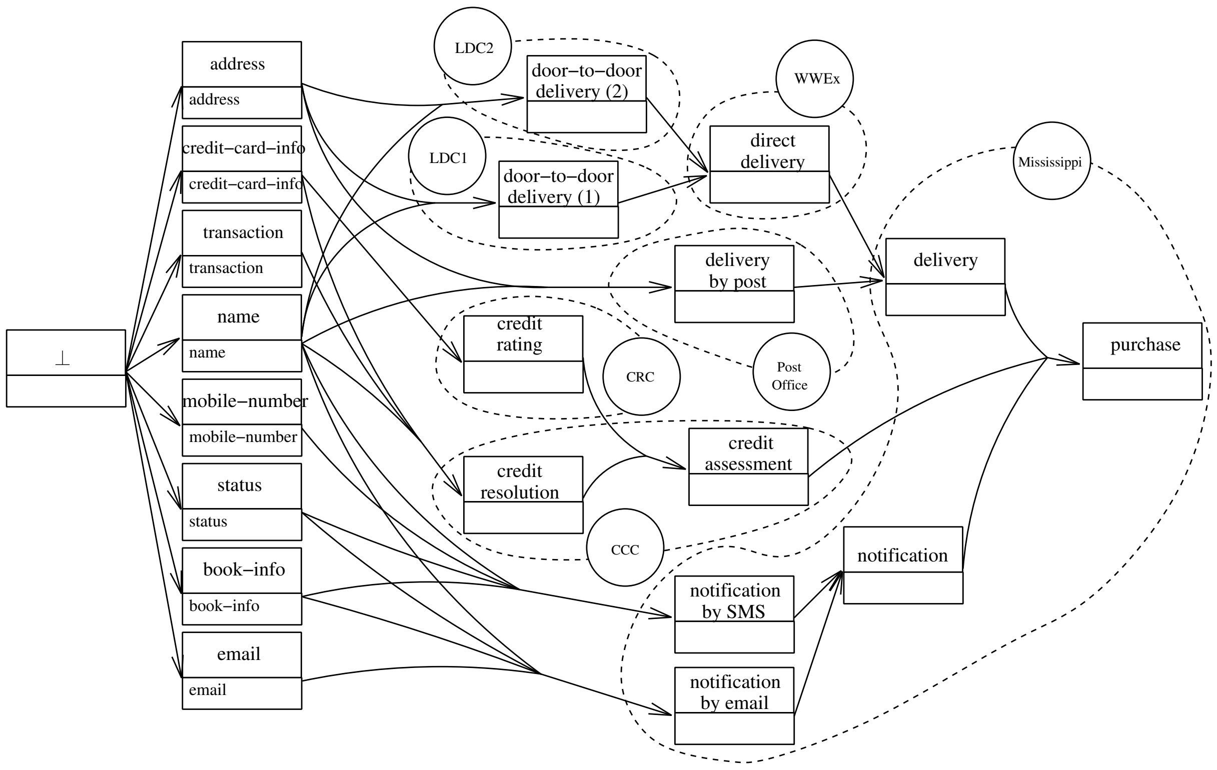

Figure 2: Purpose DAG

partners involved in the business process.[3]

Finally, in order to determine the process that discloses the minimum cost set of information, we extend purpose DAGs with the data items needed to satisfy a purpose and the privacy penalty assigned to each data item by customers. The idea is to create a node for each data item and link it to the purposes that require it. To accomplish this, we add to the purpose DAG $n + 1$ nodes where n is the number of data items occurring in the database schema. Then, each purpose is associated with the data items needed to achieve it, minus data items needed to achieve its sub-purposes:

- if a purpose node has no incoming decomposition arcs, we link to the purpose the data items needed to fulfill it with decomposition arc $\langle X, t\rangle$ where X is the set of data items and t the purpose node;
- if node t has already an incoming decomposition arc $\langle X', t\rangle$, this is replaced by the decomposition arc $\langle X \cup X', t\rangle$.

Then, each data item node is linked to the last node, *source node*, with a decomposition arc $\langle\{\perp\}, t\rangle$, where $\perp$ is the source node and t is a data item node.

Example 4 *Fig. 2 shows the purpose DAG extended within data items corresponding to the running example. Each node is composed of two parts: a purpose and the list of data items needed to fulfill it. Broken lines partition the purpose DAG in sub-purpose DAGs, and each of them represents a policy of a single*

[3]This assumption is also necessary for traditional Hippocratic databases.

partner involved in the business process. The sub-purpose DAG labeled with Mississippi corresponds to Table 3. In particular, Mississippi AND-decomposes purchase *into* delivery*,* credit assessment *whose execution is delegated to CCC, and* notification*. This means that all sub-purposes have to be reached in order to reach the root purpose. Then, the book store OR-decomposes* delivery *into* direct delivery *for which it depends on WWEx, and* delivery by post *for which it depends on Post Office. Notice that these purposes are the roots of the DAGs associated with WWEx and Post Office, respectively. Finally, Mississippi achieves* notification *either* by SMS *or* by email*. These purposes are not decomposed further and are linked instead to the data items needed to fulfill them.*

The last step of the process is to assign a weight to each decomposition arc. In our model, every decomposition arc is associated with a disclosure penalty equal to 0, except the decomposition arcs linking source node to data item nodes, and delegation arcs. The first case represents the privacy penalty to disclose data items. The latter amounts to the privacy penalty of delegating information to sub-contractors and represents the level of trust customers have towards sub-contractors. Both these assignments are given by data subjects based on their own preferences.

6 Finding a Minimal Authorization Table

When customers require a service, they are willing to disclose only information that is relevant to the service, and only to those who need it to perform their duties. In other words, customers want the process that delivers the desired service with the smallest privacy penalty. This corresponds to find the minimal decomposition path from the source node to the root purpose. Based on this path, we can build the *minimal privacy authorization table* that represents the minimum set of authorizations needed to fulfill the root purpose.

However, each customer may associate a different privacy penalty with the same data item. To cope with this requirement, we distinguish two separate phases: *Requirements Capture* and *Privacy Assessment*. The first phase is performed by the enterprise when it is designing a new business process. It needs to analyze the effect of its marketing strategies. The privacy assessment phase requires that data structures are maintained and that operations are performed on-line. This consists of dynamically maintaining reachability and minimal decomposition path when arcs are deleted, or arc weights are updated.

We list the operations required by each of these phases:

- Requirements Capture phase by business process designers: (a) initialize; (b) delete arcs, (c) add arcs, (d) adjust weights.
- Privacy Assessment phase by customers: (a) delete arcs, (b) adjust weights.

Initialization involves finding the minimum cost decomposition path of a new business process, to be suggested to customers. An enterprise could use different metrics for optimizing minimal disclosure. For example, enterprises could compute privacy penalties through statistical evidence over the customer base or could choose specific strategies depending on the different relationships with its partners. Once the enterprise has chosen a metric, algorithm MinimumCost (Fig. 3) is used to determine the minimal decomposition path. The other operations are performed by companies during the requirements capture phase, and customers during the privacy assessment phase through algorithms InsertOrDecrease (Fig. 6) and Increase (Fig. 7).

Table 9: Data Structures

Data Structure	**Type**	**Description**
$PRED[y]$	node	Pointer to the predecessor node in the minimal decomposition path from source node to simple node y.
$DISCLOSE[y]$	integer	Privacy penalty from the source node to node y.
$NEEDED[y]$	data item list	Data items needed to fulfill node y.
$TODO[y]$	integer	For simple nodes, it says if node y is reachable. For compound nodes, it is the number of simple nodes (that compound y) which are not reachable from the source.

Table 10: Algorithms for initializing and updating the minimal decomposition path

Phase	**Name**	**Input**	**Description**
I	MinimumCost		Find the minimal decomposition path for a purpose DAG.
U	InsertOrDecrease	$\langle S, t\rangle$: decomposition arc ω: weight	Update the minimal decomposition path when arcs are inserted or weight is decreased.
U	Increase	$\langle S, t\rangle$: decomposition arc ω: weight	Update the minimal decomposition path when arcs are deleted or weight is increased.

We do not include an operation for adding arcs in the privacy assessment phase. Actually, the presence of a decomposition arc corresponds to a business choice on the part of the enterprise, such as using a certain supplier or introducing additional options for delivering a service. A customer may decide not to use a particular supplier or a company's alternate option by simply ticking a check-box on a webpage.

Next, we present algorithms for finding and updating minimum cost decomposition paths and the data structures they use. A summary of the data structures is given in Table 9, while the algorithms are presented in Table 10 where I and U are respectively used for initialization and update.

7 Data Structures

In order to design efficient algorithms, we use *FD-graph* [4]. A FD-graph is essentially a labeled graph with two kinds of nodes and two kinds of edges where decomposition arcs are mapped into nodes and the two types of arcs are connecting the decomposition node to the original nodes. The following definition is based on [4].

Definition 6 *Given a purpose DAG $\mathcal{P} = \langle P, D\rangle$, let $\mathcal{S}$ be the set of source set, i.e., $\mathcal{S} = \{Z|$ there exists a decomposition arc $\langle Z, i\rangle \in D\}$. The* FD-graph *of $\mathcal{P}$ is a labeled graph $G(\mathcal{P}) = \langle P_s \cup P_c, A_{or} \cup A_{and}\rangle$, where:*

1. *$P_s \equiv P$ is a set of* simple nodes*;*
2. *P_c is the set of* compound nodes *which is in a bijective relationship with $\mathcal{S}$. If $Z \in \mathcal{S}$ is a source set then z will denote the corresponding compound node, and any simple node x_i in the source set Z will be called a* component node *of the compound node z;*

3. *$A_{or} \subseteq P_c \times P_s = \{(z,x)|\langle Z,x\rangle \in D\}$ is the set of edges referred to as* OR-edges*, in a bijective relationship with D;*

4. *$A_{and} \subseteq P_s \times P_c = \{(x_i,z)|z \in P_c$ and $x_i \in Z\}$ is the set of edges referred to as* AND-edges*, connecting any compound node to its components.*

In the sequel, we will use x and t for simple nodes, z and s for compound nodes, and y for either simple or compound nodes.

A decomposition arc is represented by a compound node with an outgoing OR-edge and one or more incoming AND-edges. Essentially, OR-edges represent a choice in selecting a decomposition arc, while AND-edges identify purposes belonging to the source set of a decomposition arc.

There is a one-to-one correspondence between the decomposition arcs of a given purpose DAG $\mathcal{P}$ and OR-edges of the corresponding FD-graph $G(\mathcal{P})$. Consequently, if a decomposition arc of $\mathcal{P}$ has weight, this is associated with the corresponding OR-edge. We represent FD-graphs as adjacency lists where all OR-edges outgoing from a node z are stored in $OUT_{or}(z)$ and all AND-edges outgoing from x in $OUT_{and}(x)$. We will also need a list of all incoming AND-edges $IN_{and}(z)$ into a compound node z and the list of incoming OR-edges $IN_{or}(x)$ into a simple node x.

Next, we present the data structures used in the algorithms. A summary is shown in Table 9.

$PRED[x]$: is used to retrieve the minimal decomposition path. The idea is to store for each simple node x, the incoming OR-edge belonging to the minimal decomposition path (*predecessors* [12]), i.e., which OR choice has been made for each node. Essentially, variable $PRED[x]$ points to the last node in the minimal decomposition path from source node $\perp$ to simple node x, otherwise, if there is no path from $\perp$ to x, it is equal to nil.

$DISCLOSE[y]$: represents the privacy penalty of the minimal decomposition path from $\perp$ to node y. For every node, the privacy penalty is initialized to infinity (∞) except for $\perp$ that is initialized to 0.

$NEEDED[y]$: maintains the data items needed to fulfill purpose y. At the beginning, for every node y, $NEEDED[y] = \emptyset$ except for the nodes associated to a data item where it contains the data item itself. In the algorithms we use $\uplus$ to indicate multi-set union.

$TODO[y]$: indicates how far we have progressed in the construction of the hibernator from $\perp$ to y. A node y is visited if the value of $TODO[y]$ is equal to 0. For any simple node x, $TODO[x]$ is initialized to 1, and for any compound node z (with components $x_1, \ldots, x_q$), $TODO[z] = q$.

8 Algorithm

The MinimumCost algorithm (Fig. 3) is based on ideas from [4] and is essentially a variant of Dijkstra classical minimum spanning tree algorithm [12]. It constructs the minimum cost decomposition path to the root purpose by working bottom up from the $\perp$ node.

Algorithm MinimumCost uses a priority[4] queue PQ whose elements have the form $(c_t, i_t, \langle s,t\rangle)$ where $\langle s,t\rangle$ is an OR-edge, and c_t and i_t denote, respectively, the privacy penalty and the list of data items that would be associated with the node t if the minimal decomposition path from $\perp$ would reach t through the edge $\langle s,t\rangle$. The queue PQ is used to sort the nodes that should be analyzed with respect

[4]Lowest data required in, first out.

```
Algorithm MinimumCost
Output:
    $DISCLOSE[y]$ : integer;
    $NEEDED[y]$ : data_item_multi-set;
    $TODO[y]$ : integer;
    $PRED[x]$ : node;
begin
    make-$PQ$-empty;
    for each {OR-edge} $\langle \perp, x \rangle \in OUT_{or}(\perp)$ do
        $PQ$-insert($\omega_{\langle \perp, x \rangle}$, $\{x\}$, $\langle \perp, x \rangle$);
    while not $PQ$-isempty do begin
        $PQ$-extract($c_t$, $i_t$, $\langle s, t \rangle$);
        if $TODO[t] \neq 0$ then begin
            $TODO[t] := 0$;
            $DISCLOSE[t] := c_t$;
            $NEEDED[t] := i_t$;
            $PRED[t] := s$;
            for each {AND-edge} $\langle t, z \rangle \in OUT_{and}(t)$ do begin
                decrement($TODO[z]$);
                if $TODO[z] = 0$ then begin
                    $DISCLOSE[z] := \sum_{\langle x,z \rangle \in IN_{and}(z)} DISCLOSE[x]$
                    $NEEDED[z] := \biguplus_{\langle x,z \rangle \in IN_{and}(z)} NEEDED[x]$
                    for each {OR-edge} $\langle z, x \rangle \in OUT_{or}(z)$ do
                        if $TODO[x] \neq 0$ then
                            $PQ$-insert($\omega_{\langle z,x \rangle} + DISCLOSE[z]$, $NEEDED[z]$, $\langle z, x \rangle$);
                end
            end
        end
    end
end
```

Figure 3: Algorithm MinimumCost

to the cost for reaching them. The algorithm starts by adding to the priority queue all initial data nodes with the corresponding privacy penalty and the node itself as identifier of the data item. The weight of the edges, i.e. the privacy penalty, is specified by the customer in his preferences (column 1 of Table 11). The algorithm extracts from the queue PQ the node t with minimum priority c_t which is assumed to be the privacy penalty of the minimal decomposition path from $\perp$ to t. This is the shortest path for t. Thereby, all AND-edges $\langle t, z \rangle$ are analyzed. For each compound node z, $TODO[z]$ is decreased, and, if it is equal to 0, we have computed the minimal decomposition path for all nodes of the source set. Then, we can compute the privacy penalty of z and add all OR-edges outgoing from z to PQ. The algorithm terminates when PQ is empty.

The output of the MinimumCost algorithm ($DISCLOSE$ and $NEEDED$) allows to build the minimum cost decomposition path, including the list of data items required by the corresponding process.

Example 5 *Table 11 reports the value of data items and delegation steps that Mississippi uses to initialize the business process. It prefers to deliver books using a delivery company because this method is safer and faster. Further, it prefers to notify via SMS because most customers are unwilling to disclose their email. Fig. 4 shows the purpose DAG presented in Fig. 2 where the discarded alternatives and data*

Table 11: User Preferences

Data Item	Cost	Delegation	Cost
name	1	CCC	2
address	5	CRC	4
email	7	WWEx	2
mobile number	2	LDC_1	2
credit-card-info	10	LDC_2	3
transaction	5	Post Office	5
book-info	2		
status	3		

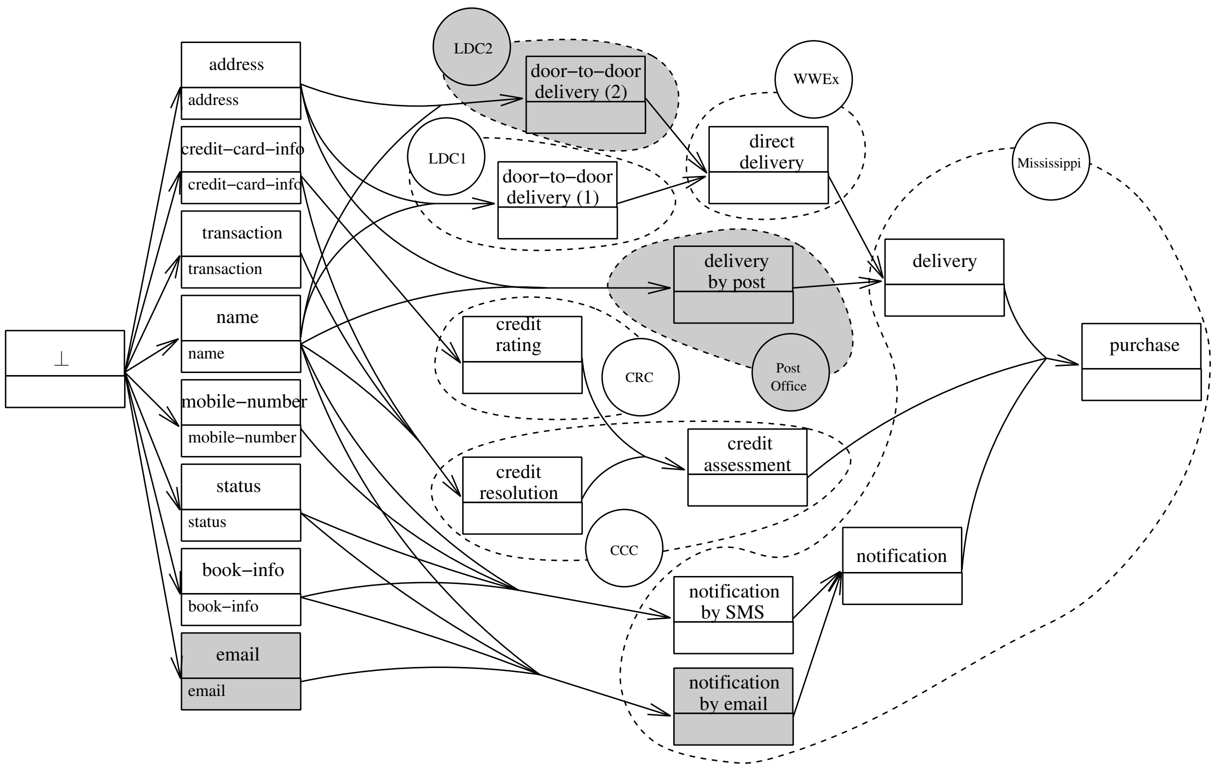

Figure 4: Minimum Decomposition Path (dark areas are not considered)

items unnecessary for delivering the service are shaded. The white nodes belong to the minimum cost decomposition path derived from the preferences of the customer. Essentially, this path represents (in form of purpose DAG) the process with the smallest privacy penalty for achieving purpose purchase.

The following results are parametric over the data structure used for managing multi-sets and priority queues. So in the sequel, we indicate by $t_{MU}(i)$ the cost for performing a multi-set union over a domain with at most i elements and by $t_{PQ}(|P|)$ the cost of managing insertion and deletion of elements in a priority queue with at most $|P|$ elements. Efficient algorithms exists for both structures [9]. For example, one can use Fibonacci heaps which have an amortized costs of $\mathcal{O}(1)$ for insertion and other operations.

Theorem 1 *Let i be the number of private data items. The algorithm terminates in $\mathcal{O}(|P| \times t_{MU}(i) \times t_{PQ}(|P|))$.*

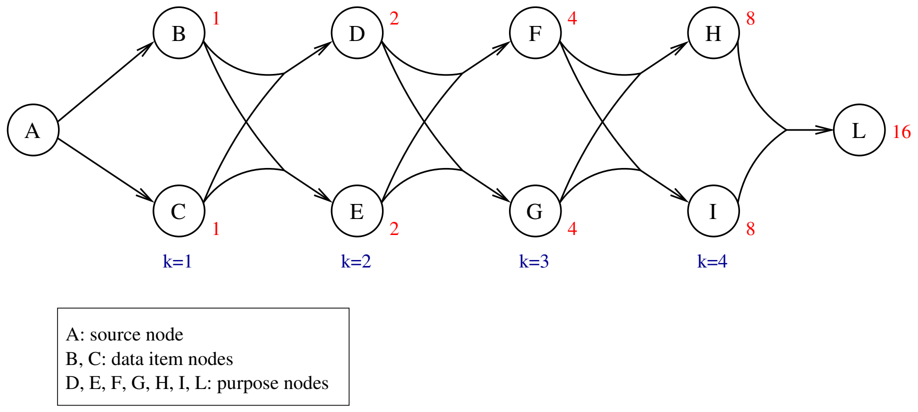

Figure 5: Concatenation of lists

Proof. First we note that each simple node is processed only once by the main algorithm because of the $TODO[t] \neq 0$ test. Also the privacy penalty of all compound nodes is computed only once (otherwise there would be a simple node in the corresponding source set that has been computed twice).

Therefore, the corresponding insertion and extraction of each outgoing OR-edge is only done once. This will amount to a $\mathcal{O}(d + p)$ cost which must be multiplied by the amortized cost of managing the priority queue, i.e., $t_{PQ}(d + p)$. The calculation of the disclosure penalty of compound nodes requires to access the source set and this would amount to an overall cost of $\mathcal{O}(a)$ because the penalty is only computed once. However, each operation requires to perform a multi-set union and therefore as a result the algorithm has a cost $\mathcal{O}(|P| \times t_{MU}(i) \times t_{PQ}(|P|))$. $\square$

Unfortunately, there are pathological cases of purpose DAG where using a simple concatenation of lists would not be adequate as it would lead to an exponential blow up of *NEEDED*. Figure 5 shows one of such cases.

Fortunately, such cases make no sense from a business perspective (and a customer should be wary of such an entangled business decomposition).

Therefore, we can conclude that

1. in the general case one must use an efficient data structure for multi-sets such as those listed in [9];

2. in all practical case the structure of a purpose DAG is essentially a tree if we do not consider data item layer. In that case a simple list concatenation is sufficient thus yielding the cost of $\mathcal{O}(a \cdot i)$. Even this upper bound is largely meaningless from practical cases as we obtain it in the case of a purpose DAG (actually tree) only made of AND-edges, whereas our working assumption is that companies wants to provide the same service in many ways (and therefore their purpose has significantly) many OR-edges.

Theorem 2 *The algorithm computes correctly the minimal privacy penalty from $\perp$ to any other node in the purpose DAG.*

Proof. The proof is by induction on the construct of the minimal decomposition path.

Table 12: Default Privacy-Authorizations Table

purpose	**table**	**attributes**	**authorized-users**
purchase	customer	{name, address, mobile-number, credit-card-info}	{ Mississippi }
purchase	order	{transaction, book-info, status}	{ Mississippi }
delivery	customer	{name, address}	{ Mississippi }
direct delivery	customer	{name, address}	{ WWEx }
door-to-door delivery	customer	{name, address}	{ LDC$_1$ }
credit assessment	customer	{name, credit-card-info}	{ CCC }
credit assessment	order	{transaction}	{ CCC }
credit scoring	customer	{credit-card-info}	{ CRC }
credit resolution	customer	{name, credit-card-info}	{ CCC }
credit resolution	order	{transaction}	{ CCC }
notification	customer	{name, mobile-number}	{ Mississippi }
notification	order	{book-info, status}	{ Mississippi }
notification by SMS	customer	{name, mobile-number}	{ Mississippi }
notification by SMS	order	{book-info, status}	{ Mississippi }

Base case The nodes reachable by the source node in 1-step are nodes representing data items. They are all inserted in the priority queue at the beginning and, since the purpose DAG is acyclic, their value is actually equal to the cost of the edge which is the minimum possible.

Ind. case When a node t is extracted from PQ for the first time, the privacy penalty associated with it is smaller than the privacy penalty associated with any node actually occurring in PQ since PQ extracts the element with the smallest cost. Suppose now that there is a minimal hyperpath from $\perp$ to t that does not pass through $\langle s, t\rangle$ but has a smaller privacy penalty. By inductive hypothesis every node y with $TODO[y] = 0$ has been reached by a minimum decomposition path so the "better" path for t must necessarily pass through some node y with $TODO[y] = 0$. So let $\langle s', t'\rangle$ be an edge of the minimal hyperpath for t that starts from a nodes with $TODO[s] = 0$. This arc has been inserted in the PQ but the privacy penalty of t' must be smaller than t, contradiction. This of course works because all edges have a non-negative privacy penalty. □

A similar result can be proved for the set of needed data items.

The minimal decomposition path is then used to build the minimal privacy authorization table where external recipients are instantiated by the corresponding authorized users. This ensures that

1. customers disclose information only if a path exists, and
2. granted authorizations are the minimum cost set necessary to fulfill the service.

If an enterprise has already defined a privacy authorization table, this latter table can be compared with the new one to verify whether it does not disclose more information than needed.

Example 6 *Table 12 shows the privacy authorization table derived from minimum cost path in Fig. 4. For example, Mississippi will notify the status of the order by SMS. Therefore, Mississippi is authorized to access customers' mobile phone numbers (but not their email addresses) for this purpose (line 11). LDC$_1$ can access data only for door-to-door delivery, and so WWEx for direct delivery. Mississippi is also entitled to access those data in order to guarantee the delivery (essentially because it needs them to* *pass them onwards to LCD_1 and WWEx). CRC is authorized to access only the data needed for credit scoring, while CCC can only access data needed for credit resolution and credit assessment.*

Differently from Hippocratic systems, we define a privacy authorization table for each customer. We argue that, by using the same authorization table for every customer, one does not properly implement the notion of minimal disclosure since such table does not take into account individual customer preferences.

Example 7 *Mississippi requires both email address and mobile phone number for notifying the status of the order (Table 3). Thus, if a customer wishes to buy a book, he has to authorize the book store to access such information (Table 6) when just one of these data items is necessary and sufficient to achieve the purpose of notification (Table 12).*

Moreover, Agrawal's proposal introduces additional unnecessary authorizations into privacy authorization tables (Tables 6 and 7). Actually, it allows a supplier to access data for fulfilling a purpose the supplier has already delegated to some sub-contractors. On the other hand, the notion of minimal disclosure implies that only those able to deliver a service should be entitled to access the data. This corresponds to the need-to-know principle for which data subjects want to ensure that their data are not delegated to recipients that do not need it. Therefore, in our scenario, Mississippi should not be entitled to access credit card information for credit assessment or shipping address for direct delivery as the minimal privacy authorization table shows (Table 12). In practice, these data should be channelled directly to the corresponding partners. A sub-optimal solution would be that this information is forwarded by Mississippi to the appropriate partners and then immediately deleted.

9 On-the-fly Updates

When a customer changes his preferences, we would like to avoid re-computing the minimum cost decomposition path from scratch after each change, but rather reuse the old solution as much as possible.

The problem of dynamically updating the purpose DAG can be essentially divided in two distinct classes:

- adding a new decomposition arc or decreasing the privacy penalty of an existing decomposition arc;
- deleting an existing decomposition arc or increasing the privacy penalty of an existing decomposition arc.

In the remainder of the section, we present the algorithms for updating the minimal decomposition path for each class of operations.

One problem of on-line procedures is to represent the FD-graph corresponding to the purpose DAG. In the case of off-line procedures, all simple and compound nodes are known a priori. On the contrary, in on-line procedures one has to take into account that new compound nodes can be inserted. Thus, when a decomposition arc is considered, one has to check whether the source set of the decomposition arc to be introduced corresponds to an existing compound node in the FD-graph. To this end, we use the function Compound that (1) returns the compound node s corresponding to source set S, if such compound node already exists, and (2) otherwise creates it and performs any necessary initialization such as computing the $DISCLOSE$ value and the $NEEDED$ multiset of data items from the corresponding values of the simple nodes in S. Once again efficient algorithms for finding a set from a pre-defined domain are well known [9].

```
Procedure InsertOrDecrease($\langle S, t\rangle$: decomposition arc, $\omega$: weight);
begin
  $s :=$ Compound$(S)$;
  if there exists no OR-edge $\langle s, t\rangle \in OUT_{or}(s)$
    then insert $\langle s, t\rangle$ into $OUT_{or}(s)$;
  $\omega_{\langle s,t\rangle} := \omega$;
  make-PQ-empty;
  PQ-insert($\omega_{\langle s,t\rangle} + DISCLOSE[s], NEEDED[s], \langle s, t\rangle$);
  while not PQ-isempty do begin
    PQ-extract($c_t, i_t, \langle s, t\rangle$);
    if $c_t \leq DISCLOSE[t]$ then begin
      $DISCLOSE[t] := c_t$;
      $NEEDED[t] := i_t$;
      $PRED[t] := s$;
      for each {AND-edge} $\langle t, z\rangle \in OUT_{and}(t)$ do begin
        $DISCLOSE[z] := \sum_{\langle x,z\rangle \in IN_{and}(z)} DISCLOSE[x]$
        $NEEDED[z] := \biguplus_{\langle x,z\rangle \in IN_{and}(z)} NEEDED[x]$
        for each {OR-edge} $\langle z, x\rangle \in OUT_{or}(z)$ do
          $PQ$-insert($\omega_{\langle z,x\rangle} + DISCLOSE[z], NEEDED[z], \langle z, x\rangle$);
      end
    end
  end
end
```

Figure 6: Procedure InsertOrDecrease

9.1 Insert or Decrease

The procedure InsertOrDecrease (Fig. 6) maintains the minimum cost decomposition path when new decomposition arcs are inserted or the cost of an existing decomposition arc is decreased.

Such changes can introduce a new minimal path and therefore we propagate them bottom-up until a root of some subpath remains unchanged. This algorithm exploits the observation that the previous minimal decomposition path does not change its value (if a new arc is inserted or the decreased arc is not in that path), or even improves it (if the decreased arc is in the current minimal path). Accordingly, we can use the value of $DISCLOSE$ to separate the useful from the not useful computation in the same way that we used the $TODO$ variable in the off-line algorithm for the minimum cost.

Firstly, the procedure takes as input a decomposition arc $\langle S, t\rangle$ and its privacy penalty and determines the compound node s corresponding to the source set S using function Compound. If such a node already exists, its penalty is updated; otherwise, the new decomposition arc and its penalty are inserted in the DAG. Notice that the first case corresponds to decreasing the penalty of existing decomposition arcs, while the second corresponds to adding a new decomposition arc. The idea is to verify whether the decomposition arc yields a decomposition path that improves the old penalty. If it is the case, the decomposition arc is considered; otherwise the algorithm terminates since the minimum cost path does not change.

9.2 Increase

When the customer increases the privacy penalty of decomposition arcs we use the algorithm Increase (Fig. 7) to build the new minimal decomposition path. It can also be used for arc deletions by setting the

```
Procedure Increase($\langle S, t\rangle$: decomposition arc, $\omega$: weight);
begin
  $s :=$ Compound$(S)$;
  $\delta := \omega - \omega_{\langle s,t\rangle}$;
  $\omega_{\langle s,t\rangle} := \omega$;
  if $PRED[t] = s$ then begin
      $DISCLOSE[t] := DISCLOSE[t] + \delta$;
      for each {OR-edge} $\langle z, t\rangle \in IN_{or}(t)$ do
          $PQ$-insert($\omega_{\langle z,t\rangle} + DISCLOSE[z], NEEDED[z], \langle z, t\rangle$);
      while not PQ-isempty do begin
          PQ-extract($c_t, i_t, \langle s, t\rangle$);
          if $c_t \leq DISCLOSE[t]$ then begin
              $DISCLOSE[t] := c_t$;
              $NEEDED[t] := i_t$;
              $PRED[t] := s$;
              for each {AND-edge} $\langle t, z\rangle \in OUT_{and}(t)$ do begin
                  $DISCLOSE[z] := \sum_{\langle x,z\rangle \in IN_{and}(z)} DISCLOSE[x]$
                  $NEEDED[z] := \biguplus_{\langle x,z\rangle \in IN_{and}(z)} NEEDED[x]$
                  for each {OR-edge} $\langle z, x\rangle \in OUT_{or}(z)$ do
                      if $PRED[x] = z$ then begin
                          $DISCLOSE[x] := DISCLOSE[z] + \omega_{\langle z,x\rangle}$;
                          for each {OR-edge} $\langle s, x\rangle \in IN_{or}(x)$ do
                              $PQ$-insert($\omega_{\langle s,x\rangle} + DISCLOSE[s], NEEDED[s], \langle s, x\rangle$);
                      end
              end
          end
      end
  end
end
```

Figure 7: Procedure Increase

weight equal to infinity (∞).

The idea behind the algorithm is that if the decomposition arc whose weight has been increased does not belong to the minimum cost decomposition path, the minimum path does not change. Here the tricky bit is the following: if the modified arc belongs to the current minimum path before the increase then it might no longer be in the minimum path after the increase. Therefore, we must update also the value of the $DISCLOSE$ variable to take into account the fact that this is no longer the best-value but the best-so-far.

The procedure takes as input an existing decomposition arc $\langle S, t\rangle$ and its updated value and determines the compound node s corresponding to the source set S by using function Compound.[5] If the arc belongs to the decomposition path, the value of the $DISCLOSE$ penalty is updated to take into account that it has "worsened". Then, the siblings of the arc $\langle s, t\rangle$ – the arcs having node t as head – are examined in order to check whether one of them might improve the best-so-far $DISCLOSE$ value. When one of those arcs yields a decomposition path that improves the best-so-far penalty, all its AND-edges $\langle t, z\rangle$ are analyzed, because they must have changed (at least node t in the source set has changed $DISCLOSE$ value). Then, we start analyzing the outgoing OR-edges and check whether some of those also belong to the previous minimal decomposition path. In this case, we apply the same reasoning that we applied before: we update

[5]We assume that the compound node exists since we are considering only weight increase and arc deletion.

Table 13: Alice's User Preferences

Data Item	**Cost**	**Delegation**	**Cost**
name	1	CCC	2
address	5	CRC	4
email	4	WWEx	∞
mobile number	20	LDC$_1$	∞
credit-card-info	10	LDC$_2$	∞
transaction	5	Post Office	5
book-info	2		
status	3		

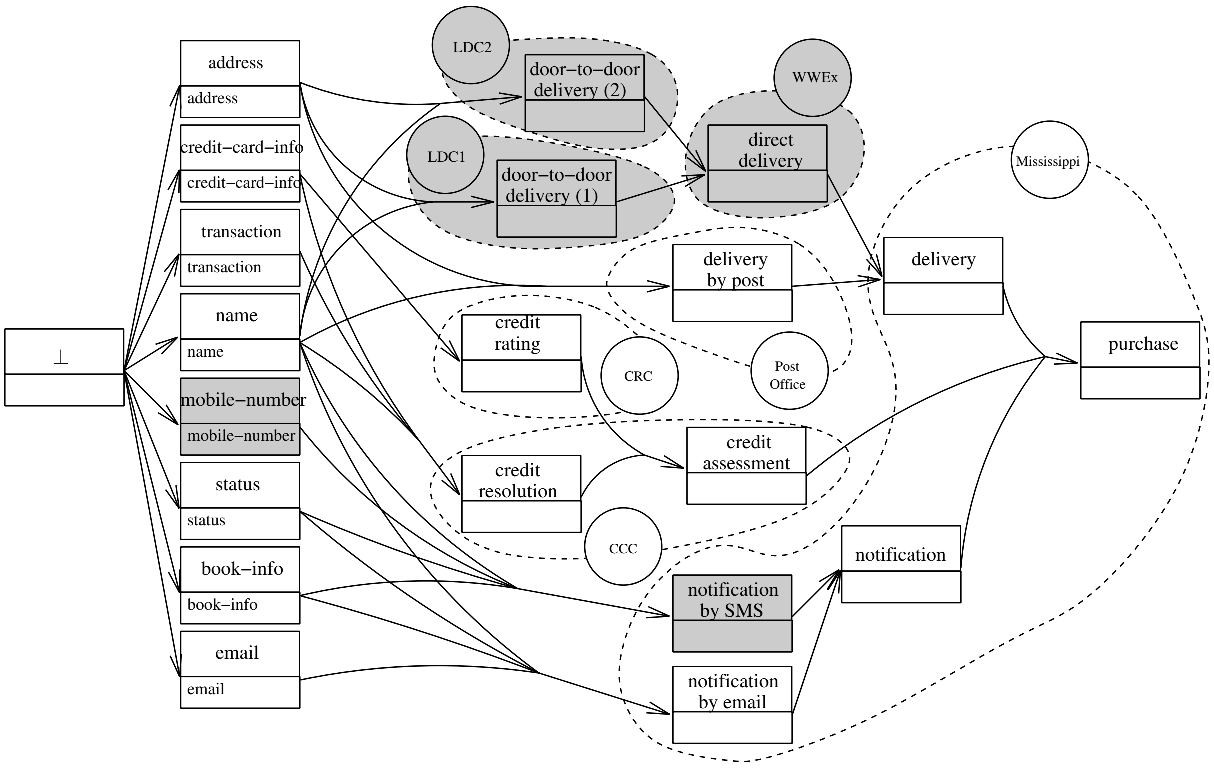

Figure 8: Minimum Decomposition Path (dark areas are not considered)

the $DISCLOSE$ variable to the best-so-far value that we have just computed and insert in the PQ all the siblings of the selected node to check whether any of them can improves the current tentative path.

Example 8 *Alice wants to buy some books, but she does not agree with the default customer preferences offered by Mississippi. In particular, she prefers to receive books by post because she does not trust to give her address to delivery companies after a bad experience with a local delivery company. To this end, she defines the cost of delegating information to WWEx and to local delivery companies equal to infinity*[6] *(∞). Further, she does not have a business mobile phone and she would prefer to not give her personal mobile phone number. In contrast, she has a very good anti-spam filter and is therefore willing to communicate her email address. Thus, she sets the cost of providing her mobile phone number equal*

[6]This corresponds to deleting the decomposition arc.

Table 14: Alice's Privacy-Authorizations Table

purpose	**table**	**attribute**	**authorized-users**
purchase	customer	{name, address, email, credit-card-info}	{ Mississippi }
purchase	order	{transaction, book-info, status}	{ Mississippi }
delivery	customer	{name, address}	{ Mississippi }
delivery by post	customer	{name, address}	{ Post Office }
credit assessment	customer	{name, credit-card-info}	{ CCC }
credit assessment	order	{transaction}	{ CCC }
credit scoring	customer	{credit-card-info}	{ CRC }
credit resolution	customer	{name, credit-card-info}	{ CCC }
credit resolution	order	{transaction}	{ CCC }
notification	customer	{name, email}	{ Mississippi }
notification	order	{book-info, status}	{ Mississippi }
notification by email	customer	{name, email}	{ Mississippi }
notification by email	order	{book-info, status}	{ Mississippi }

to 20 and the cost of providing her email address equal to 4. Table 13 summarizes Alice's preferences and Fig. 8 shows the purpose DAG representing the business process where the discarded alternatives and data items unnecessary for delivering the service are shaded. The white nodes represent the minimal decomposition path computed with respect to Alice's preferences.

As mentioned earlier, the minimal decomposition path is used to build the minimal privacy authorization table.

Example 9 *Table 14 reports the privacy authorization table corresponding to the minimal decomposition path in Fig. 8. It shows that Mississippi cannot access Alice's mobile phone number for notification and that WWEx and local delivery companies cannot access any data; only the Post Office is entitled to access her data for delivering the purchased books.*

10 Related Works

The last years have seen an increasing attention on privacy-aware technologies and mechanisms for the negotiation of private information between customers and companies. These are particularly critical for transactions carried out over the web [13] where the "customer" negotiating the private information might not be a human but rather a software system.

Among work centered on the notion of purpose, LeFevre et al. [18] enhance Hippocratic databases with mechanisms for enforcing queries to respect privacy policies stated by an enterprise and customer preferences. In essence, they propose to enforce the minimal disclosure principle by providing mechanisms to data owners that control who can access their personal data and for which purpose.

To support the negotiation of private information, the World Wide Web Consortium (W3C) proposed the Platform for Privacy Preferences (P3P) [10]. This standard provides mechanisms that allow customers to check web site privacy policies before they disclose their personal data to the site. Another mechanism for negotiation is presented by Tumer et al. [27]. Enterprises specify which information is mandatory for achieving a service and which is optional, while customers specify the type of access for each part of their personal information: free (i.e., the access is granted without conditions), limited (i.e., the access is granted only if the enterprise has defined as mandatory that part of information), or not given (i.e., the access is never granted). Then, the framework matches enterprise policies with customer preferences. If mandatory information is not given by a customer, the framework verifies if alternative strategies stated by the enterprise match customer preferences in order to reach an agreement with the customer.

Mechanisms for enforcements are proposed by Karjoth et al. [5, 17]. The Enterprise Privacy Authorization Language (EPAL) [5] enables an enterprise to exactly formalize the privacy policies that shall be enforced within the enterprise itself. However, these proposals do not provide mechanisms for enforcing the minimal disclosure principle. In Byun et al. [7], the Role-Based Access Control model is extended by introducing the notion of purpose and a purpose management model. Similarly to our approach, they introduce purpose hierarchies in order to reason on access control. However, their hierarchies are based on the principles of generalization and specialization and are not expressive enough to support complex strategies defined by enterprises.

An alternative approach proposed by Thuraisingham [26] introduced the notion of privacy constraints. In this proposal, every role that users can play is associated with a privacy level. A role at a certain privacy level can access only data at or below that privacy level. Hence, the privacy problem focuses on determining whether individual privacy can be violated given a set of constraints which assign a privacy level to data. Yusuda et al. [28] define a purpose-oriented access control model for controlling information flow. Essentially, an information flow is defined as “legal” only if such information are used for a certain purpose.

A policy itself may be sensitive because from the analysis of the disclosed policies an unauthorized user may infer sensitive information. Following this observation, some approaches propose to protect not only personal information, but also policies themselves [24].

11 Conclusion

This paper presents a framework for supporting the management of privacy-sensitive data within business processes provided by virtual organizations. In this setting, a company can deliver a service to a customer in many ways and by relying on many different partners. The selection of the partners and the identification of a particular plan to fulfill a purpose can potentially be done on-the-fly. Yet, such selection should so be driven by the customer’s desire to minimize the exposure of privacy sensitive data.

In particular, our approach improves Hippocratic database systems by providing (1) a framework to model business processes that span across multiple partners and make use of AND/OR purpose decomposition hierarchies, (2) efficient algorithms and data structures to select during the design phase the business plan and the business partners that fulfill the desired purpose while minimizing the private information that is requested from customers, (3) modified on-line algorithms that allow a customer to change its privacy preferences and penalties on-the-fly and to recompute the selection of the business plan and the business partners according the customer’s own criteria for minimal disclosure.

There are clearly open issues. First, one may wish to describe the internal structure of organizations in order to have a fine grained model that could also capture the number of units and the number of individuals accessing the information. Second, there is the need to extend current transaction models for virtual organizations [22] in order to ensure that no information is disclosed until all partners have committed to the delivery of their part of the business plan using only the information that the customers are willing to make available to each of them. Our algorithms guarantee that the business process can be completed but a certified and transactional commitment may be desirable in some cases. The (global) commitment should ensure that if the customer provides the requested personal information then the virtual organization can actually deliver the service. These issues are left as future work.

Acknowledgments This work was partly supported by the projects RBNE0195K5 FIRB-ASTRO, 016004 IST-FP6-FET-IP-SENSORIA, 27587 IST-FP6-IP-SERENITY, 2003-S116-00018 PAT-MOSTRO, 1710SR-B/P PAT-STAMPS.

References

[1] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. of SIGMOD'03*, pages 86–97. ACM Press, 2003.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic Databases. In *Proc. of VLDB'02*, pages 143–154. Morgan Kaufmann, 2002.

[3] G. Ausiello, P. G. Franciosa, and D. Frigioni. Directed Hypergraphs: Problems, Algorithmic Results, and a Novel Decremental Approach. In *Proc. of ICTCS'01*, volume 2202 of *LNCS*, pages 312–327. Springer-Verlag, 2001.

[4] G. Ausiello, R. Giaccio, G. F. Italiano, and U. Nanni. Optimal Traversal of Directed Hypergraphs. Technical Report TR-92-073, The International Computer Science Institute (ICSI), September 1992.

[5] M. Backes, B. Pfitzmann, and M. Schunter. A Toolkit for Managing Enterprise Privacy Policies. In *Proc. of ESORICS'03*, volume 2808 of *LNCS*, pages 162–180. Springer-Verlag, 2003.

[6] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: An Agent-Oriented Software Development Methodology. *JAAMAS*, 8(3):203–236, 2004.

[7] J.-W. Byun, E. Bertino, and N. Li. Purpose Based Access Control of Complex Data for Privacy Protection. In *Proc. of SACMAT'05*, pages 102–110. ACM Press, 2005.

[8] C. L. Chang and J. R. Slage. An admissible and optimal algorithm for searching AND/OR graphs. *Artif. Intell.*, 2:117–128, 1971.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, second edition, 1990.

[10] L. Cranor, M. Langheinrich, M. Marchiori, and J. Reagle. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. W3C Recommendation, Apr. 2002.

[11] Y. Desmedt and Y. Wang. Maximum flows and critical vertices in and/or graphs. In *Proc. of COCOON'02*, pages 238–248. Springer-Verlag, 2002.

[12] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerical Mathematics*, 1:269–271, 1959.

[13] E. Ferrari and B. M. Thuraisingham. Security and Privacy for Web Databases and Services. In *Proc. of the 9th Int. Conf. on Extending Database Technology*, volume 2992 of *LNCS*, pages 17–28. Springer-Verlag, 2004.

[14] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2-3):177–201, 1993.

[15] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling Security Requirements Through Ownership, Permission and Delegation. In *Proc. of RE'05*, pages 167–176. IEEE Press, 2005.

[16] C. Handy. Trust and the virtual organization. *Harvard Business Review*, 73:40–50, 1995.

[17] G. Karjoth, M. Schunter, and M. Waidner. Platform for Enterprise Privacy Practices: Privacy-enabled Management of Customer Data. In *Proc. of PET'02*, volume 2482 of *LNCS*, pages 69–84. Springer-Verlag, 2002.

[18] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. J. DeWitt. Limiting Disclosure in Hippocratic Databases. In *Proc. of VLDB'04*, pages 108–119. Morgan Kaufmann, 2004.

[19] A. Martelli and U. Montanari. Additive AND/OR Graphs. In *Proc. of IJCAI'73*, pages 1–11. Morgan Kaufmann, 1973.

[20] F. Massacci and N. Zannone. Privacy is Linking Permission to Purpose. In *Proc. of the 12th Int. Workshop on Sec. Protocols*, 2004.

[21] N. J. Nilsson. *Problem solving methods in AI.* McGraw-Hill, 1971.

[22] M. P. Papazoglou. Web Services and Business Transactions. *World Wide Web: Internet and Web Inform. Sys.*, 6:49–91, 2003.

[23] S. Sahni. Computationally related problems. *SIAM J. on Comp.*, 3(4):262–279, 1974.

[24] K. E. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis. Protecting Privacy during On-line Trust Negotiation. In *Proc. of PET'02*, volume 2482 of *LNCS*, pages 129–143. Springer-Verlag, 2002.

[25] R. Sebastiani, P. Giorgini, and J. Mylopoulos. Simple and minimum-cost satisfiability for goal models. In *Proc. of CAiSE'04*, volume 3084 of *LNCS*, pages 20–35. Springer-Verlag, 2004.

[26] B. Thuraisingham. Privacy constraint processing in a privacy-enhanced database management system. *Data and Knowl. Eng.*, 55(2):103–236, 2005.

[27] A. Tumer, A. Dogac, and H. Toroslu. A Semantic based Privacy Framework for Web Services. In *Proc. of ESSW'03*, 2003.

[28] M. Yasuda, T. Tachikawa, and M. Takizawa. Information Flow in a Purpose-Oriented Access Control Model. In *Proc. of ICPADS'97*, pages 244–249. IEEE Press, 1997.