Minimal Disclosure in Hierarchical Hippocratic Databases with Delegation*

Fabio Massacci[1], John Mylopoulos[1,2], and Nicola Zannone[1]

[1] Department of Information and Communication Technology,
University of Trento - Italy
{massacci, zannone}@dit.unitn.it
[2] Department of Computer Science,
University of Toronto - Canada
jm@cs.toronto.edu

Abstract. Hippocratic Databases have been proposed as a mechanism to guarantee the respect of privacy principles in data management. We argue that three major principles are missing from the proposed mechanism: hierarchies of purposes, delegation of tasks and authorizations (i.e. outsourcing), and the minimal disclosure of private information.

In this paper, we propose a flexible framework for the negotiation of personal information among customers and (possibly virtual) enterprises based on user preferences when enterprises may adopt different processes to provide the same service. We use a goal-oriented approach to analyze the purposes of a Hippocratic system and derive a purpose and delegation hierarchy. Based on this hierarchy, effective algorithms are given to determine the minimum set of authorizations needed for a service. In this way, the minimal authorization table of a global business process can be automatically constructed from the collection of privacy policy tables associated with the collaborating enterprises. By using effective online algorithms, the derivation of such minimal information can also be done on-the-fly by the customer wishing to use the services of a virtual organization.

1 Introduction

Since the early works on privacy protection in statistical databases [1], privacy research has gained momentum. Changes in the landscape of legislation around the world, and growing consumer attention to the issue have changed attitudes towards security and privacy concerns for database systems. This matches with a substantial body of research on approaches for managing the negotiation of personal information among customers and enterprises [2,3,18,20].

At the basis of every solution for the exchange between enterprises and customers, there is the principle of transparency. Transparency means that when enterprises store data about customers they should disclose to customers which data is collected and how it is used, i.e., for what *purpose* data is maintained. Starting from the landmark

* This work has been partially funded by the IST programme of the EU Commission, through an FET under the IST-2001-37004 WASP project, by the FIRB programme of MIUR under the RBNE0195K5 and RBAU01P5SS projects, by MOSTRO and SMTPPs projects of PAT.

S. De Capitani di Vimercati et al. (Eds.): ESORICS 2005, LNCS 3679, pp. 438–454, 2005.

proposals for Hippocratic databases [3], most privacy-aware technologies use purpose as a central concept around which privacy protection is built. For the transparency requirement, enterprises should declare in their privacy policies the purpose for which data is collected, who can receive it, the length of time the data can be retained and the authorized users who can access it. Looking at such policies customers would be able to understand how their personal data will be used and, in case they agree, disclose them.

Transparency is not the only principle, and another important notion which goes hand in hand with transparency is the notion of minimal disclosure, as defined in the US Privacy Act of 1974 and the EU Directives on Privacy in 1995. This principle requires enterprises to maintain only the information necessary to fulfill the purpose for which it has been collected. The principle of minimal disclosure seems to be easily satisfiable. A company must simply ask the necessary data and leave other useful but unnecessary fields as optional. We all experience this business practice when filling in a web form.

However, enterprises are able to provide their services in different ways, and each different method could require different data. For example, banks may deliver bank statements by email and by regular post. Depending on the method, customers should provide their shipping address or email to the bank. Asking for both addresses as compulsory would clearly violate the principle of minimal disclosure.

If we consider these decisions, the burden of choice is on the human who must decide what to do on the basis of his/her personal feeling of trust of the enterprises. But this is very difficult for complex tasks, where there are many ways to deliver the service. The situation is worse if we consider dynamic coalitions, such as those that might be soon available with Web Services and Business Processes for Web Services. On the server side, we might not have a single enterprise, but rather a host of partners participating in a business process. Further, companies may outsource a large part of data processing to external supplier which on their own may do a similar process.

In some cases, the client process may even no longer be a human deciding to fill an email field with her business email or a freshly created Yahoo address but rather a software client. A software process needs automatic procedures for making such a judgment on the basis of some general criteria provided by the user.

Classical privacy-aware systems such as Hippocratic Databases do not consider these issues of delegation, minimality and their automatic treatment. In this paper we show how to address them.

1.1 The Contribution of This Paper

This paper presents a flexible framework for automatically deriving the minimum set of authorizations needed to achieve a service (i.e., the minimal privacy authorization table) from the enterprise privacy policy (privacy policy table) by determining the minimum set of data needed to fulfill required services based on users preferences and the partners entitled to access the data.

Following goal-oriented security requirements engineering approaches [9], we propose to analyze the purposes behind the design of a Hippocratic system, and organize them in hierarchal manner through AND- and OR-decompositions and delegation. Further, we extend that hierarchy by associating to purposes the data needed to accomplish them. Once customers have given a weight to each piece of data, one can determine

the minimum set of data for fulfilling the root purpose with respect to user preferences. Reasoning procedures for the fulfillment of users' requirements by different solutions have been already investigated in goal oriented requirements engineering [10,19]. However, their solution is not adequate for our purposes, as it is tailored to off-line analysis by the system designer and not to on-the-fly selection by the system user.

In order to have more efficient algorithms, we represent purpose and delegation hierarchies with hypergraphs [4,5]. Based on this data structure, we provide algorithms for finding a minimal decomposition path that represents the process that uses the minimum set of information to fulfill a purpose, and for efficiently updating it when users change the cost of data items or choose among the alternatives that an enterprise offers for achieving the required service. Then, this path is used to determine the minimum set of authorizations needed to achieve a service.

The remainder of the paper is structured as follows. Next (§2) we introduce a scenario used as running example throughout the paper. We then provide (§3) a brief description of Hippocratic databases. Then, we introduce purpose DAGs in order to represent purpose hierarchies (§4) and discuss how to build a purpose DAG from a Hippocratic database system (§5). Next (§6) we present algorithms for finding and updating the minimum cost path. Finally, we discuss related works and conclude the paper with some directions for future work (§7).

2 A Running Example

Our scenario is a revised version of the case study proposed by Agrawal et al. [3].

Mississippi is an on-line bookseller who needs to obtain certain personal information to perform purchase transactions. This information includes name, shipping address, and credit card number. Mississippi views purchase (the root-level "purpose" for its service) as a three-step process: credit assessment, delivery, and notification. Delivery can be done by direct delivery or by post, while notification can be done by email or by fax. Depending on the method of notification, Mississippi needs either email or fax information.

Mississippi relies on Worldwide Express (WWEx) for shipping books. WWEx is a delivery company that offers a global network of specialized services – transportation, international trade support and supply chain services. WWEx also needs personal information to delivery books for Mississippi. This information includes customer name and shipping address. In turn, WWEx 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 $\mathrm{LDC}_1, \ldots, \mathrm{LDC}_n$ 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 to obtain some information for providing credit assessment. This information includes customer's name and credit card number, and the transaction between Mississippi and the customer. For making credit decisions, CCC wants a credit rating[1]. For this, CCC depends on the Credit Rating Company (CRC). CRC uses statistics to summarize past experience so that predictive analysis can be used to generate

[1] Credit rating is a method for interpreting the content of a credit report.

Table 1. Database Schema

table	attribute
customer	purpose, customer-id, name, address, email, fax-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 }

a rating for the customer. Based on the rating, CCC can decide to accept or not the customer transaction.

3 A Primer on Hippocratic Databases

Hippocratic databases [2,3] use *purpose* as a central concept and consider it as a "special" attribute occurring in every tables forming the database and associated with each piece of data stored in the database.

Example 1. Table 1 shows the schema of two tables, *customer* and *order*, that store the personal information collected by Mississippi.

Then, for each purpose and for each data item stored in the database, we have:

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

Purpose, external recipients, authorized users, and retention period are stored in the database with respect to the metadata schema defined in Table 2 [3]. Specifically, the above information is split into separate tables: external-recipients and retention period are in *Privacy-Policies* Table (PPT), while authorized-users in *Privacy-Authorizations* Table (PAT). The purpose is stored in both of them. PPT contains the privacy policies of the enterprise, while PAT contains the access controls policies that implement the privacy policy and represents the actual disclosure of information. In particular, PAT is created from PPT by instantiating each external recipient with the corresponding users. Therefore, Hippocratic systems define a PAT for each PPT. These tables are equal for every customer, and so they do not appreciate individual user preferences.

Example 2. According the PPT of Mississippi, it can access both email and fax number for notifying the status of the order. WWEx, Post Office, and all LDCs can access customer data for direct delivery, delivery by post and door-to-door delivery, respectively. These authorizations match exactly the policies declared in the corresponding PPTs.

Further examples for the PPT of each partner involved in the business process and the corresponding PATs in our running example can be found in [14].

Before users disclose their information, the *Privacy Constraint Validator* is used to verify whether user preferences match the privacy policy of the enterprise. In this way, Hippocratic DBs implement the consent principle. When queries are submitted to the database, the system answers only queries for which the purpose is equal to that for which data has been stored. Further, Hippocratic DBs do not disclose information for purposes different from those for which the owner of the information have previously give the consent. Thus, Hippocratic DBs implement, respectively, the limited use and disclosure principles. To enforce the retention principle, Hippocratic DBs use the *Data Retention Manager* which deletes data items when their retention period is expired.

The limited collection principle requires that enterprises collect the minimum set of data needed to fulfill the purpose for which data is stored. Hippocratic DBs use three components to implement such principle: *Access Analysis*, that identifies for each purpose which data never occurs in query answers; *Granularity Analysis*, that determines the granularity of the required information; *Minimal Query Generation*, that designs queries that disclose the minimum set of data needed for fulfilling a certain purpose.

4 Hierarchy and Delegation of Purposes

Hippocratic systems are an elegant and simple solution but do not allow for dynamic situations that could arise with web services and business process software. In such settings, enterprises may provide services in many different ways and may delegate the execution of parts of the service to third parties. This is indeed the case of a virtual organization based on business process for web service where different partners explicitly integrate their efforts into one process. This affects mainly the creation of the PAT.

Agrawal et al. [3] propose to split a purpose into multiple purposes and then store them in the database. In this way, we lose the relation among a purpose and its sub-purposes. Karjoth et al. [11] use a directory-like notation to represent purpose hierarchies. However, this notation does not distinguish if a sub-purpose is derived by AND or OR decomposition, and consequently cannot be used to reason about the fulfillment of the root purpose. Additionally the same sub-purpose may be part of different purposes. This distinction is important from the perspective of minimality of information. For example, providing *both* an email address and a physical address might be needed to provide the password for access to the tracking service and the actual shipping of goods and those purposes may be both necessary (AND) to obtain a certain higher level goal. However, in other cases only one of them could be necessary (OR). Therefore, requiring both of them would be a violation of the minimality principle.

Our approach is based on traditional goal analysis [15], and consists of decomposing purposes into sub-purposes through an AND/OR refinement. If purpose p is AND-decomposed (respectively, OR-decomposed) into sub-purposes $p_1, \ldots, p_n$, then all (at least one) of the sub-purposes must be satisfied for satisfying p. The idea is to represent purpose hierarchies with hypergraphs [4,5], and we will call them *purpose directed acyclic graphs* (or *purpose DAGs*, for short).

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).*

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* sub purpose DAG *of $\mathcal{P}$. This is denoted by $\mathcal{P}' \subseteq \mathcal{P}$.*

The enterprise-wide privacy policies is derived by looking at the Hippocratic database of each partner involved in the business process and merging them into a single purpose DAG. Therefore, purpose DAGs can be recognized as the outcome of a process of refinements of goals and delegation of tasks in security requirements modeling methodologies [9]. Fig. 1 shows an example of purpose DAG. Each node is composed by two parts: a purpose identifier and the list of data items needed to fulfill the purpose. Broken lines partition the purpose DAG in sub purpose DAG, and each of them represents the policies of a single enterprise, and so purposes on the broken line can be seen as services whose execution is delegated to other suppliers.

Definition 3. *Let $\mathcal{P} = \langle P, D\rangle$ be a purpose DAG, $X \subseteq P$ be a non-empty subset of purposes, and y be a purpose in P. A* decomposition path *$\mathcal{D}_{X,y}$ is a set of decomposition arcs $D' \subseteq D$ such that either $y \in X$ or there exists a decomposition arcs $\langle Z, y\rangle \in D'$ and there are decomposition paths $\mathcal{D}_{X,z} \in D'$ for each $z \in Z$.*

Essentially, a decomposition path represents a possible process through which an enterprise can fulfill a root purpose. Our goal is to decide which is the process with the "minimum privacy penalty" to fulfill the root purpose with respect to the user's preferences. This can be performed through a quantitative analysis. In order to support quantitative analysis, we need to introduce the notion of weighted purpose DAG.

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

Since decomposition paths have a complex structure, different ways can be used to measure the cost of the same decomposition path. Depending on the weight measure, the problem can be polynomially tractable [8] or NP-hard [6,7,17]. The problem of finding a minimal cost hyperpath in a directed hypergraph is shown to be NP-hard when the cost of a hyperpath is the sum of the weights of its hyperarcs [4,5]. By making the cost function additive, Martelli and Montanari [13] were able to formulate a polynomial time algorithm for AND/OR graphs. For additive cost functions, the cost of one edge is counted as many times as it is traversed. Additive cost functions are also considered in hypergraph approaches that find optimal hyperpaths in polynomial time [4,5].

For our purposes, we use an additive cost function. We believe that additive measures are the ones that capture best the intuitive way in which we might wish to protect our privacy. In a nutshell, if the same datum is disclosed N times, then the cost of these disclosures is N, rather than 1. After all, the more a datum is used, the more it is likely that it might be compromised, or the more it is likely to end up in companies not so privacy-aware. The more our data are tossed back and forth the less happy we are.

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

1. *if* $y \in X$, *then* $dp(X, y) = 0$
2. *if path* $\mathcal{D}_{X,y}$ *has root* $\langle Z, y\rangle$ *with subpath* $\mathcal{D}_{X,z_1}, \ldots, \mathcal{D}_{X,z_k}$, *then* $dp(X, y) = \omega_{\langle Z,y\rangle} + \sum_{z_i \in Z} dp(X, z_i)$.

5 From Hippocratic DBs to Purpose DAG

We now have the machinery to construct a purpose DAG when orchestrating a business process composed by many different partners (each with its own Hippocratic DB). The construction is sketched below.

- For each supplier PPT, purposes are analyzed through a goal refinement process, and so they are structured with respect to AND/OR decomposition. These purpose DAGs are circumscribed by a broken line and labeled with the supplier's name.
- Once we have a DAG for each supplier, we build the DAG representing the privacy policy of the entire business process by merging them.
- Then each purpose is associated with the data items directly needed to achieve the purpose itself (data items needed to achieves its sub-purposes are linked directly to sub-purposes).

Merging is done by looking at the external-recipients field in every PPT: when the external-recipients field is not empty, we connect that purpose with the corresponding purpose (with the same name) occurring in the DAG associated with the supplier that is an instance of some external recipient. If there is more than one instance for the same external recipient, we create a fictitious node and OR decompose it into a number of nodes equal to the number of possible instances. This is also what happens if we have 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 possible. To support this process, we assume a common ontology among all the actors involved in the purpose DAG.[2]

The last step takes into account the data items we need to satisfy a purpose and the privacy penalty assigned to each data item by users. The idea is to create a node for each data item and link it to the purposes that directly requires it. So, we add to the purpose DAG $n + 1$ nodes where n is the number of data items. Then, 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. Otherwise, if node t has already an incoming decomposition arc $\langle X', t\rangle$, this is replaced by the decomposition arc $\langle X \cup X', t\rangle$. We link to each data item nodes the last node, *source node*, with arc $\langle \perp, t\rangle$, where $\perp$ is the source node and t is a data item node.

Example 3. Fig. 1 shows the purpose DAG extended within data items corresponding to the running example. Each purpose DAG on a broken line represents the hierarchical

[2] This assumption is also necessary in Hippocratic database systems. If external recipients of data could assign a semantics to a purpose that is different from the semantics assigned by the Hippocratic database owner we could as well eliminate the entire tagging process and provide all data with purpose "do-what-you-please".

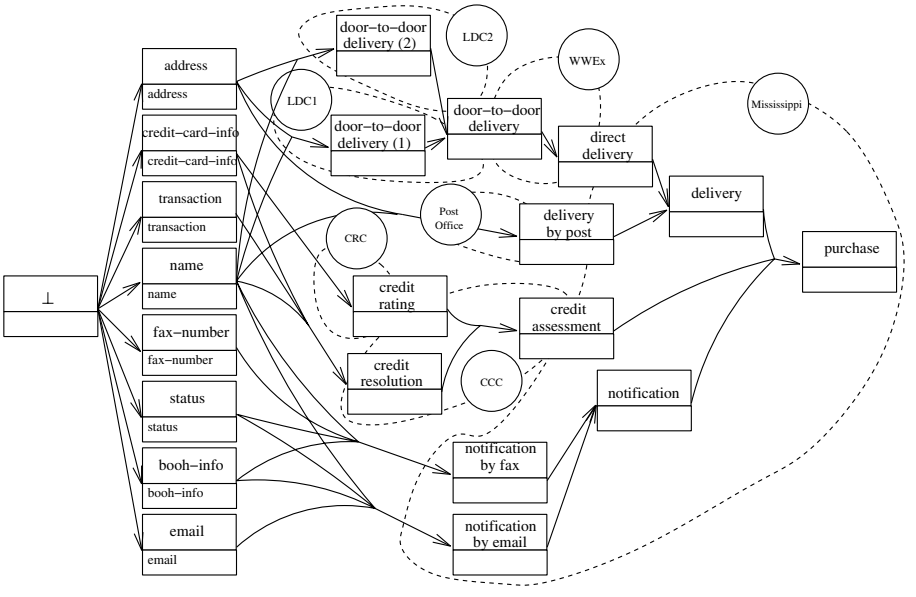

Fig. 1. Purpose DAG

model of the privacy policies concerning a partner. In particular, Mississippi AND-decomposes *purchase* into *delivery*, *credit assessment* whose execution is delegated to CCC, and *notification*. Thus, all sub purposes have to be reached in order to reach the root purpose. Then, the store OR-decomposes *delivery* into *direct delivery* for which it depends on WWEx, and *delivery by post* for which it depends on Post Office. These purposes are the root of the DAGs associated with WWEx and Post Office. Finally, the store achieves *notification* either *by fax* or *by email*. These purposes are not further on decomposed, and so are linked to the data items needed to fulfill them.

Every decomposition arc has disclosure penalty equal to 0, except the decomposition arcs joining source node and data item nodes, and delegation arcs. In the first case, the disclosure penalty corresponds to the cost of perceived disclosure of data. The latter represents the disclosure penalty to delegate information. Both these assignments are given by data owners with respect to their own preferences. In particular, weight on delegation edges from one supplier to the sub-suppliers can be defined by asking the users to specify the level of trust they feel about sub-suppliers.

6 Minimum Cost Algorithms

Customers do not want to disclose more data than needed to get the desired service. This corresponds to finding the minimal decomposition path from the source node to the root purpose. This path can be used to build the minimal PAT that represents the minimum set of authorizations for fulfilling the root purpose. A key observation is that such computation cannot in general be done by the company providing the service

Table 3. Data Structures

Data Structure	Type	Description
$LAST[y]$	node	Pointer to the last node in the minimal 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.

once and for all customers: customers may associate a different privacy penalty to the provision of the same data item. Therefore, they are interested in finding the minimum information cost for fulfilling the root purpose with respect to their own preferences. The computation of minimal preferences is essentially a dynamic on-line process.

In order to design efficient algorithms for dynamic evaluation of privacy preferences, we use *FD-graph* [5] whose definition is given below.

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 $\langle Z, i\rangle \in D$ and $|Z| > 1\}$. 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 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 z_i in the source set Z will be called a* component node *of compound node z;*
3. *$A_{or} \subseteq (P_c \times P_s) \cup (P_s \times P_s) = \{(z, x)|\langle Z, x\rangle \in D\}$ is the set of edges referred to as* OR-edges*, in bijective relationship with D;*
4. *$A_{and} \subseteq P_s \times P_c = \{(z_i, z)|z \in N_c$ and $z_i \in Z\}$ is the set of edges referred to as* AND-edges*, connecting any compound node to its components*

Essentially, a decomposition arc is represented by a compound node with a leaving OR-edge and one or more incoming AND-edges. The OR-edge corresponds to the OR choice of selecting the decomposition arc. Once the decomposition arc is selected, all purposes in its source set must be fulfilled. 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 $G(\mathcal{P})$. If a decomposition arc of $\mathcal{P}$ has a weight, this is associated to the corresponding OR-edge. FD-graphs can be implemented by maintaining adjacency lists where all OR (AND) edges leaving a node y are organized in $L_{or}(y)$ ($L_{and}(y)$).

When we design a system we can distinguish two phases, namely *Requirements Capture phase* and *Privacy Assessment phase*. Each of these phases involves some operations: the Requirements Capture phase requires an initialization phase and support for deleting arcs, adding arcs, increasing weights and decreasing weights, while the Privacy Assessment phase requires support for deleting arcs and increasing weights.

Next, we present the data structures used in the algorithms. A summary of such data structures is shown in Table 3. In order to retrieve the minimal decomposition path, the idea is to store for each simple node y, the incoming decomposition arcs belonging to the minimal decomposition path (*backward pointers* [5]) by using $LAST[y]$. This points to the last node in the minimal decomposition path from source node $\perp$ to simple node y, otherwise, if there is no path from $\perp$ to y, it is equal to nil.

Table 4. Algorithms for initializing and updating the minimal decomposition path

Phase	Name	Input	Description
I	MinimumCost		Find the minimal decomposition path for a purpose DAG.
I	ScanMC	t: node x: simple-node	Scan OR-edges and update priority queue. Called by MinimumCost.
U	WeightIncrease	$\langle X, y\rangle$: decomposition arc ω: weight	Update the minimal decomposition path when arcs are delated or weight is increased.
U	ScanWI	t: node x: simple-node	Scan OR-edges and update priority queue. Called by Insert and WeightIncrease.

The privacy penalty of the minimal decomposition paths from $\perp$ to any other simple or compound node y is stored in $DISCLOSE[y]$. For every node, the privacy penalty is initialized to infinity (∞). The list of data item needed to fulfill a purpose y is stored in the variable $NEEDED[y]$. At the beginning, for every node y, $NEEDED[y] = \emptyset$ except for the nodes associated to a data item where the list contains the data item itself. The symbol $\uplus$ is used to represent concatenation of lists. Finally, the variable $TODO[y]$ is used to store if there is a path 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 $z_1, \ldots, z_q$), $TODO[z]$ is initialized to $\sum_{k=1}^{q} TODO[z_k]$.

In the remainder of the section, we present some algorithms for finding and updating the minimum cost decomposition path. A summary of such algorithms is given in Table 4 where I and U are respectively used for initialization and update.

6.1 Initialization

Initialization refers to find the minimum cost decomposition path for a new purpose DAG. The following algorithms are based on [5] and are essentially a variant of Dijkstra classical minimum spanning tree algorithm. The algorithms are described in the following, while the pseudocode is given in Fig. 2 and 3.

Algorithm MinimumCost uses a priority[3] 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 are, respectively, the privacy penalty and the list of data items associated with the node t. The algorithm inserts as a first element in the priority queue the item $(0, \emptyset, \langle \perp, \perp\rangle)$. Then, repeatedly, 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. Thereby, all OR-edges outgoing from t are scanned by procedure ScanMC, 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 the privacy penalty of the minimal decomposition path from $\perp$ to z is computed. Then, all OR-edges outgoing from z are scanned by procedure ScanMC. Procedure ScanMC aims at analyzing OR-edges $\langle t, x\rangle$: if the ingoing node x is not already visited, the procedure inserts it in the priority queue; otherwise, the penalty of x is updated if and only if edge $\langle t, x\rangle$ improves the old penalty associated with x.

[3] Lowest data required in, first out.

```
Algorithm MinimumCost
Output:
   DISCLOSE[y] : integer;
   NEEDED[y] : data_item_list;
   TODO[y] : integer;
   LAST[y] : node;
begin
   make-PQ-empty;
   PQ-insert(0, ∅, ⟨⊥, ⊥⟩);
   TODO[⊥] := 0;
   while PQ-nonempty do begin
      PQ-extract(C_t, I_t, ⟨s, t⟩);   {extract from the queue PQ the node t with minimum priority C_t}
      DISCLOSE[t] := C_t;
      NEEDED[t] := I_t;
      LAST[t] := s;
      for each {OR-edge} ⟨t, x⟩ ∈ L_or(t) do ScanMC(t, x);
      for each {AND-edge} ⟨t, z⟩ ∈ L_and(t) do begin
         decrement(TODO[z]);
         if TODO[z] = 0          {If node z is reached the privacy penalty of the path from ⊥ to z is computed}
            then begin
                 DISCLOSE[z] := Σ_{z_i ∈ z} DISCLOSE[z_i]
                 NEEDED[z] := ⊎_{z_i ∈ z} NEEDED[z_i]
                 for each {OR-edge} ⟨z, x⟩ ∈ L_or(z) do ScanMC(z, x);
            end
      end
   end
end
```

Fig. 2. Algorithm MinimumCost

```
Procedure ScanMC(t: node; x: simple-node);
begin
   C_{t,x} := ω_⟨t,x⟩ + DISCLOSE[t];
   I_{t,x} := NEEDED[t];
   if TODO[t] = 1                          {check if node t has been previously visited}
      then begin
           decrement(TODO[t]);               {if not, node t is marked as reached
           PQ-insert(C_{t,x}, I_{t,x}, ⟨t, x⟩);   and arc ⟨t, x⟩ is inserted in PQ}
      end
      else if C_{t,x} < C_x                  {otherwise, PQ is update only if arc ⟨t, x⟩ improves minimal path}
           then PQ-decrease(C_{t,x}, I_{t,x}, ⟨t, x⟩);
end
```

Fig. 3. Procedure ScanMC

Example 4. Defining default preferences, Mississippi gives a value on data items and delegation steps (Table 5). It prefers to deliver books by using a delivery company because this method is more secure and faster. Further, it prefers to notify by fax. Fig. 4 shows the minimum cost path. Comparing it with Fig. 1, we can see that email does not occur anymore since fax has a lower penalty. Also the DAGs labeled with Post Office and LDC_2 are no longer considered since the sum of the penalties associated with WWEx and LDC_1 is lower than those associated with Post Office and LDC_2.[4]

[4] The penalty for delegating data to LDC_2 includes the trust level associated with WWEx.

Table 5. User Preferences

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

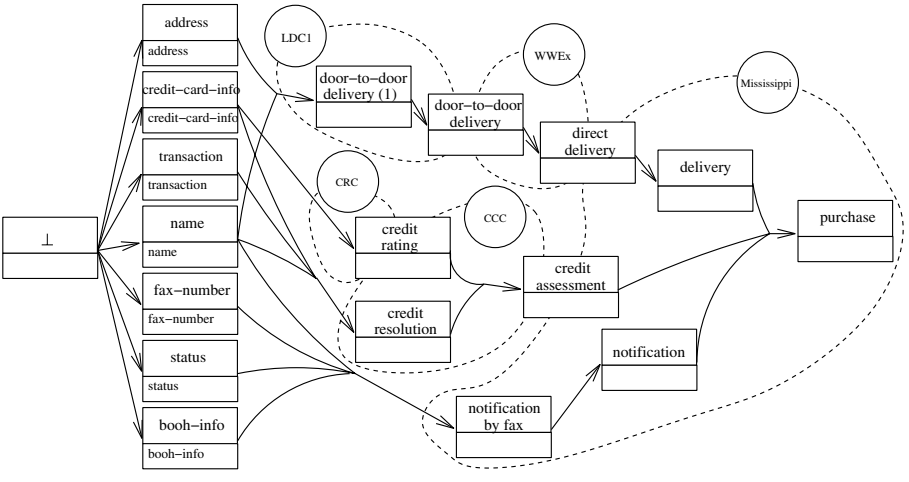

Fig. 4. Minimum Decomposition Path

It is possible to prove, as done in [5] that

1. a node y is marked (i.e., $TODO[y] = 0$) if it is reachable from the source node;
2. the algorithm computes correctly the minimal privacy penalty from $\perp$ to any other node in the purpose DAG;
3. the algorithm terminates in linear time in the size of the purpose DAG.

Every purpose can be seen as a business process. Business processes can be combined, and the "new" process can be seen as an atomic process. Atomic processes follow the ACID properties [16] that guarantee that all participants will see the same outcome: in case of success all services make the results of their operation permanent by commitment, otherwise all services undo all operations they have requested and data is not disclosed. Thus, to guarantee consistent and reliable execution, we should check if the minimal path exists. This path is then used to build the PAT where external recipients are instantiated by the corresponding authorized users. This ensures that a user discloses all information needed to fulfill the service only if a path exists and that disclosed information is the minimum cost set of data necessary to fulfill the service.

Example 5. Mississippi is authorized to notify the status of the order only by fax, and so it can collect only data related to that purpose for notification. LDC_1 can access only data needed for door-to-door delivery, and so WWEx for direct delivery. In turn, Mississippi is entitled to access those data for achieving delivery. Moreover, CRC is authorized to access only data need for credit rating and CCC for credit resolution. Then, CCC can access only those data for performing credit assessment. Finally, Mississippi is entitled to collect data for achieving purchase in accordance with those allowed for its sub purposes. As shown in Fig. 4, Mississippi cannot access customer email.

A comparison among the PATs derived by two approaches is given in [14].

6.2 On-the-Fly Update of Customer Privacy Preferences

Both requirements capture and privacy assessment phases require to update the solution when weights are modified. In particular, the privacy assessment phase requires that data structures are maintained and that operations are performed on-line. The idea is to reuse the valid part of the old solution as much as possible.

The problem of dynamically updating the purpose DAG can be essentially divided in two distinct classes depending on the update operations that are possible:

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

For sake of generality both possibilities must be considered when devising the theory but we argue that most practical implementations will only have to cope with the second type of updates. Indeed the presence of a decomposition arc corresponds to a business choice done by the enterprise (such as using a supplier). A customer may surely decide *not* to use a particular supplier, without further ado than ticking a checkbox on the web. However, *adding* a supplier or a partner to a business process is a procedure that can be conceived for very dynamic virtual business coalitions, and requires to solve problems (system integration, commercial agreement, legal liabilities etc.) that go well beyond the comparatively simple issue of privacy preferences. So we leave to the technical report [14] the details of the procedure that maintains the minimum cost path when new arcs are inserted or the cost of an existing arc is decreased.

In the case the customer increases the privacy penalty of decomposition arcs, we use algorithm WeightIncrease to build the new minimum cost decomposition path. The pseudocode is given in Fig. 5 and 6. The idea is that if the decomposition arc does not belong to the minimum cost decomposition path, this path does not change since we are analyzing only weight increase and arc deletion. If the decomposition arc belongs to the minimal path, we examine the other decomposition arcs having node t as head. To this end, we use the function *backward* B_{or} where, given a node x, $B_{or}(x) = \{h \in D | x = head(h)\}$. Essentially, $B_{or}(x)$ is the set of incoming decomposition arcs of x. Any time a decomposition arc that does not belong to the minimum cost path is found, it is pruned. The procedure WeightIncrease can be simply re-used for the case of arc deletion by defining the weight equal to infinity (∞).

Example 6. Alice, a Mississippi's customer, does not agree with default user preferences given by Mississippi. In particular, she prefers to receive books by post because

```
Procedure WeightIncrease(⟨X, y⟩: decomposition arc, ω: weight);
begin
      if |X| = 1
      then x := the single element of X;
      else x := Compound(X);
  if LAST[y] = x then begin                {arc ⟨x, y⟩ is considered only if it belongs to minimal path}
      DISCLOSE[y] := ω + DISCLOSE[x];
      for each {OR-edge} ⟨s, y⟩ ∈ B_or(y) do ScanWI(s, y);
      while PQ-nonempty do begin
         PQ-extract(C_t, I_t, ⟨s, t⟩);     {extract from the queue PQ the node t with minimum priority C_t}
         DISCLOSE[t] := C_t;
         NEEDED[t] := I_t;
         LAST[t] := s;
         for each {OR-edge} ⟨t, x⟩ ∈ L_or(t) do
            if LAST[x] = t then            {arc ⟨t, x⟩ is considered only if it belongs to minimal path}
               for each {OR-edge} ⟨s, x⟩ ∈ B_or(x) do ScanWI(s, x);
         for each {AND-edge} ⟨t, z⟩ ∈ L_and(t) do begin
            c := Σ_{z_i ∈ z} DISCLOSE[z_i]
            d := ⊎_{z_i ∈ z} NEEDED[z_i]
            if c < DISCLOSE[z] then begin  {arc ⟨t, z⟩ is considered only if it improves minimal path}
               DISCLOSE[z] := c
               NEEDED[z] := d
               for each {OR-edge} ⟨z, x⟩ ∈ L_or(z) do
                  if LAST[x] = z then      {arc ⟨z, x⟩ is considered only if it belongs to minimal path}
                     for each {OR-edge} ⟨s, x⟩ ∈ B_or(x) do ScanWI(s, x);
            end
         end
      end
end
```

Fig. 5. Procedure WeightIncrease

```
Procedure ScanWI(t: node; x : simple-node);
begin
   C_{t,x} := ω_⟨t,x⟩ + DISCLOSE[t];
   I_{t,x} := NEEDED[t];
   if C_{t,x} < DISCLOSE[x]  {arc ⟨t, x⟩ is considered only if it improves minimal path}
      then if ⟨t, x⟩ ∉ PQ
         then PQ-insert(C_{t,x}, I_{t,x}, ⟨t, x⟩);
         else PQ-decrease(C_{t,x}, I_{t,x}, ⟨t, x⟩);
end
```

Fig. 6. Procedure ScanWI

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 delegation information to WWEx equal to infinity (∞). Further, she does not have a personal fax and must use her company's fax where faxes are first given to the program manager's secretary for distribution to the staff. Thus, she defines the cost of fax number equal to 20. Fig. 7 shows the minimal path computed with respect to her user preferences. The corresponding PAT [14] shows that Mississippi cannot access her fax number for notification and that WWEx and local delivery companies cannot access any of her data; only Post Office is entitled to access her data for delivering the purchased books.

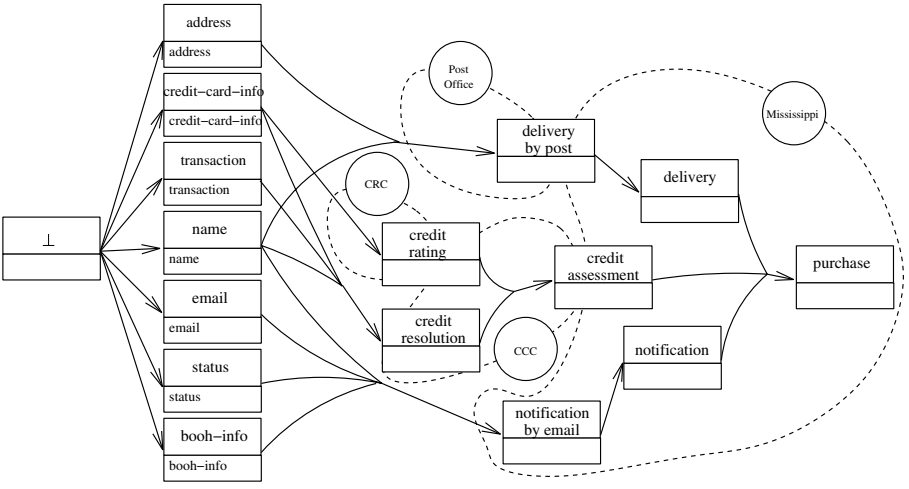

Fig. 7. Minimum Decomposition Path

7 Related Work and Conclusion

Last years have seen an increasing attention to privacy-protection technologies and the negotiation of private information between customers and companies. Tumer et al. [20] present a framework for Web Services that allows users and enterprises to automatically negotiate personal information. Each data item is defined as *Mandatory* or *Optional* by an enterprise, while users define for each part of their personal information the kind of access, namely *Free*, *Limited*, or *NotGiven*. Then, the framework matches enterprise policies with user preferences. If a mandatory input is not given by a user, enterprises can find alternative strategies in order to reach an agreement with the user.

A policy itself may be sensitive since analyzing the disclosed policies an unauthorized user may infer sensitive information. Therefore, some approaches aim not only to protect personal information, but also policies themselves. LeFevre et al. [12] provide an approach for forcing queries to respect privacy policies stated by an enterprise and users preferences. Their idea is to specify additional conditions to regulate the disclosure of information. Another approach to avoid unauthorized disclosure of sensitive information is Automated Trust Negotiation [18]. It aims to regulate iterative disclosures of credentials and requests between requesters and provider. These approaches are different from ours since we assume that information are committed only after checking that enterprise policies comply with user preferences. We argue that, if policies are not known at priori, users cannot know which data they have to provide. It may be possible that users discover that an enterprise requires more information than they (the users) consider reasonably sufficient to provide the service only when they have already disclosed part of their information.

The main contribution of this paper is a framework for deriving the minimum set of authorizations needed to provide a service by determining the minimum set of information a customer has to give. In particular, our approach provides support to Hippocratic

systems for enforcing the limited collection principle when a complex business process is analyzed and user preferences are considered. Indeed Hippocratic systems create a privacy authorization table shared by all customers. This does not allow to distinguish which particular method is used for delivering a service, and so to customize the minimum set of information. Therefore, access analysis is only able to determine which data items are never used for a purpose and, consequently, minimal query generation works on a set of information that is not minimum. Finally, our framework ensures that a user discloses all (and only) the information required by the process that uses the minimum set of information to delivery the service.

There are some issues left as future work. One of these is to introduce an actor hierarchy to model the hierarchical nature of organizations (e.g., company-division-department-individual worker). Further, customers must be assured that they are getting a complete and correct answer to their queries before delegating privacy information. To this end, we are investigating the usage of Merkle Trees to build a global certificate to be provided to the client by composing the individual certificates from the various business partners.

References

1. N. R. Adam and J. C. Worthmann. Security-control methods for statistical databases: a comparative study. *CSUR*, 21(4):515–556, 1989.
2. R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. of the 2003 ACM SIGMOD Int. Conf. on Management of Data*. ACM Press, 2003.
3. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic Databases. In *Proc. of VLDB'02*, pp. 143–154. Morgan Kaufmann, 2002.
4. G. Ausiello, P. G. Franciosa, and D. Frigioni. Directed Hypergraphs: Problems, Algorithmic Results, and a Novel Decremental Approach. In *Proc. of ICTCS'01*, *LNCS 2202*, pp. 312–327. Springer-Verlag, 2001.
5. G. Ausiello, R. Giaccio, G. F. Italiano, and U. Nanni. Optimal Traversal of Directed Hypergraphs. Technical Report TR-92-073, ICSI, September 1992.
6. C. L. Chang and J. R. Slage. An admissible and optimal algorithm for searching AND/OR graphs. *Artif. Intell.*, 2:117–128, 1971.
7. Y. Desmedt and Y. Wang. Maximum flows and critical vertices in and/or graphs. In *Proc. of COCOON'02*, pp. 238–248. Springer-Verlag, 2002.
8. G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2-3):177–201, 1993.
9. P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Requirements Engineering meets Trust Management: Model, Methodology, and Reasoning. In *Proc. of iTrust'04*, *LNCS 2995*, pp. 176–190. Springer-Verlag, 2004.
10. P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Reasoning with Goal Models. In *Proc. of ER'02*, pp. 167–181, 2002.
11. G. Karjoth, M. Schunter, and M. Waidner. Platform for Enterprise Privacy Practices: Privacy-enabled Management of Customer Data. In *Proc. of PET'02*. Springer-Verlag, 2002.
12. K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. J. DeWitt. Limiting Disclosure in Hippocratic Databases. In *Proc. of VLDB'04*, 2004.
13. A. Martelli and U. Montanari. Additive AND/OR Graphs. In *Proc. of IJCAI'73*, pp. 1–11. Morgan Kaufmann Publisher, INC., 1973.

14. F. Massacci, J. Mylopoulos, N. Zannone, Minimal Disclosure in Hierarchical Hippocratic Databases with Delegation, Technical Report DIT-05-051, Univ. di Trento 2005. Available on the web at `http://eprints.biblio.unitn.it`.
15. N. J. Nilsson. *Problem solving methods in AI*. McGraw-Hill, 1971.
16. M. P. Papazoglou. Web Services and Business Transactions. *World Wide Web: Internet and Web Inform. Sys.*, 6:49–91, 2003.
17. S. Sahni. Computationally related problems. *SIAM J. on Comp.*, 3(4):262–279, 1974.
18. K. Seamons, M. Winslett, and T. Yu. Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation. In *Proc. of NDSS'01*, pp. 109–125. IEEE Press, 2001.
19. R. Sebastiani, P. Giorgini, and J. Mylopoulos. Simple and minimum-cost satisfiability for goal models. In *Proc. of CAiSE'04*, *LNCS 3084*, pp. 20–35. Springer-Verlag, 2004.
20. A. Tumer, A. Dogac, and H. Toroslu. A Semantic based Privacy Framework for Web Services. In *Proc. of ESSW'03*, 2003.