CERIAS Tech Report 2004-52

PURPOSE BASED ACCESS CONTROL FOR PRIVACY PROTECTION IN RELATIONAL DATABASE SYSTEMS

by Ji-Won Byun and Elisa Bertino and Ninghui Li

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Purpose Based Access Control for Privacy Protection in Relational Database Systems

Ji-Won Byun
byunj@cs.purdue.edu

Elisa Bertino
bertino@cerias.purdue.edu

Ninghui Li
ninghui@cs.purdue.edu

Center for Education and Research in Information Assurance and Security
and Department of Computer Sciences
Purdue University
656 Oval Drive, West Lafayette, IN 47907

ABSTRACT

In this paper, we present a comprehensive approach for privacy preserving access control based on the notion of purpose. Purpose information associated with a given data element specifies the intended use of the data element, and our model allows multiple purposes to be associated with each data element. A key feature of our model is that it also supports explicit prohibitions, thus allowing privacy officers to specify that some data should not be used for certain purposes. Another important issue addressed in this paper is the granularity of data labeling, that is, the units of data with which purposes can be associated. We address this issue in the context of relational databases and propose four different labeling schemes, each providing a different granularity. In the paper we also propose an approach to representing purpose information, which results in very low storage overhead, and we exploit query modification techniques to support data access control based on purpose information.

1. INTRODUCTION

Current information technology enables people to carry out their business virtually at any time in any place. At the same time, it also offers the capability to watch online users' every move and to store many types of information the users reveal during their activities. Indeed, a study conducted by the Federal Trade Commission in May 2000 [7] shows that 97 percent of web sites were collecting at least one type of identifying information such as name, e-mail address, or postal address of consumers. The fact that their personal information can be collected, stored and used without their consent or awareness creates fear of privacy violation for many people. The advance of database technology has also significantly increased privacy concerns as the current database technology makes it possible to store a massive amount of data and extract various kinds of information.

Even though the direct victims of privacy violations are consumers, many enterprises and organizations are deeply concerned about privacy issues as well. Some companies, such as IBM and the Royal Bank Financial Group, use privacy as a brand differentiator [2]. By demonstrating good privacy practices, these businesses try to build solid trust and provide more confidence to customers, thereby attracting more customers. Potential lawsuits brought up by consumers and recently enacted privacy legislations also require organizations to pay attention to the management of private data.

As privacy becomes a major concern for both consumers and enterprises, many privacy protecting access control models have been proposed [2, 9, 1, 12]. We emphasize that privacy protection cannot be easily achieved by traditional access control models. Privacy policies are concerned with which data object is used for which purpose(s), rather than which user is performing which action on which data object. For example, a typical privacy policy such as "we will collect and use customer identifiable information for billing purposes and to enable us to anticipate and resolve problems with your service" does not state who can access the customer information, but states that the information can be accessed for the purposes of billing, customer service, and possibly some analysis. Another difficulty of privacy protection is that the comfort level of data usage varies from individual to individual. For example, some online consumers may feel that it is acceptable to disclose their purchase history or browsing habits in return for better service, such as site personalization [11]. Other customers, however, may believe strongly that such techniques violate their privacy.

Observing these challenges, we believe that in order to protect data privacy, the notion of purpose must play a major role in access control models and that an appropriate metadata model must be developed to support such privacy-centric access control models. In this paper, we address this goal by presenting a comprehensive approach to purpose management, which is the fundamental building block on which purpose-based access control can be developed. Our approach is based on intended purposes, which specify the intended usage of data, and access purposes, which specify the purposes for which a given data element is accessed. Both intended purposes and access purposes are specified with respect to a hierarchical structure organizing a set of purposes for a given enterprise. A key feature of our proposed model is that it also supports explicit prohibitions, thus allowing privacy officers to specify that data should not be used for a given set of purposes. We also formally define the notion of purpose compliance, which is the basis for verifying that the purpose of a data access complies with the intended purposes of the data. An important issue that we also address in this paper is the granularity of data labeling; that is, the units of data with which purposes can be associated. We address this issue in the context of relational databases and propose four different labeling schemes, each providing a different granularity. Using our approach it is

thus possible to associate a purpose (or a set of purposes) with an entire table, with each column within a table, with each tuple within a table, or with each attribute within a tuple. In the paper we also propose an approach to representing purpose information, which results in very low storage overhead. Furthermore, we exploit query modification techniques to support data filtering based on purpose information. Such techniques ensure efficient query processing even in the case of fine-grained purpose labeling. Finally, in the Appendix to the paper, we show possible extensions to SQL to support purpose management.

The remainder of this paper is organized as follows. Section 2 provides a brief overview for privacy related technologies available today. Section 3 formally defines the notion of purpose and describes its hierarchy and some relevant issues. Section 4 describes our labeling model in detail. Section 5 presents our implementation, and Section 6 provides experimental results. Section 7 suggests future work and concludes our discussion.

2. RELATED WORK

Our work is related to many different areas of private and secure data management, namely privacy policy specification, privacy-preserving data management systems and multilevel secure database systems. We now briefly survey the most relevant approaches in these areas and point out the differences of our work with respect to these approaches.

The W3C's Platform for Privacy Preference (P3P) [18] is an industry standard that intends to provide an automated method for users to gain control over the use of their personal information on web sites they visit. P3P allows web sites to encode their privacy practice, such as what information is collected, who can access the data for what purposes, and how long the data will be stored by the sites, in a machine-readable format. P3P enabled browsers can read this privacy policy automatically and compare it to the consumer's set of privacy preferences which are specified in a privacy preference language such as A P3P Preference Exchange Language (APPEL) [17], also designed by the W3C.

Even though P3P provides a standard means for enterprises to make privacy promises to their users, P3P does not provide any mechanism to ensure that these promises are consistent with the internal data processing. That is, P3P is merely a tool for making promises and does not help enterprises to keep their promises. Note that publishing an attractive P3P policy without any adequate enforcement mechanism may put an enterprise at risk of reputation damage and potential lawsuits.

The Enterprise Privacy Authorization Language (EPAL) [9] proposed by IBM is a formal language for writing enterprise privacy policies to govern data handling practices in IT systems. An EPAL policy defines lists of hierarchies of data-categories, user-categories, and purposes. User-categories are the entities (users/groups) that use collected data, and data-categories define different categories of collected data that are handled differently from a privacy perspective. Purposes model the services for which data is intended to be used. An EPAL policy also defines sets of actions, obligations, and conditions. Actions model how the data is used, and obligations define actions that must be taken by the environment of EPAL. Lastly, conditions are boolean expressions that evaluate the context. Privacy authorization rules are defined using these elements, and each rule allows or denies actions on data-categories by user-categories for certain purposes under certain conditions while mandating certain obligations.

While providing a language for specifying policies on data categories, EPAL does not provide support for linking the data categories with data stored in databases. Nor does the EPAL work address the issue of how to efficiently enforce these policies when data is accessed.

Previous work on multilevel secure relational databases [14, 4, 15, 8] also provides many valuable insights for designing a fine-grained secure data model. In a multilevel relational database system, every piece of information is classified into a security level, and every user is assigned a security clearance. Based on this access class, the system ensures that each user gains access to only the data for which he has proper clearance, according to the well known basic restrictions, the Bell-LaPadula model [3]. These constraints ensure that there is no information flow from a lower security level to a higher security level and that subjects with different clearances see different versions of multilevel relations.

A major difference of our approach with respect to multilevel secure databases is that in our approach each data element is associated with set of purposes, as opposed to a single access class. Also, the purposes form a hierarchy and can vary dynamically. These requirements are more complex than those concerning traditional multilevel secure applications. On the other hand, we are not concerned with information flow issues in this paper.

The concept of Hippocratic databases, incorporating privacy protection within relational database systems, was introduced by Agrawal et al.[1]. The proposed architecture uses privacy metadata, which consist of privacy policies and privacy authorizations stored in two tables. A privacy policy defines for each attribute of a table the usage purpose(s), the external-recipients and retention period, while a privacy authorization defines which purposes each user is authorized to use.

Recently, Lefevre et al.[12] present an approach to enforcing privacy policy in database environments. Their work focuses on ensuring limited data disclosure, based on the premise that data providers[1] have control over who is allowed to see their personal data and for what purpose. In their work, they introduce two models of cell-level limited disclosure enforcement, which are table semantics and query semantics. They also suggest an implementation based on query modification techniques.

Although their work is closely related to ours, our approach has some notable differences. First, we introduce more sophisticated concepts of purpose; i.e., purpose hierarchy. Note that our approach, even though more sophisticated, does not increase the complexity of data management, nor does it introduce much overhead. The second difference is that we support the explicit prohibition of purpose and the association of a set of purposes with a data element, which their approach does not provide. Third, we provide a comprehensive framework for purpose and data management and also suggest possible extensions to SQL, which are not considered in their work.

[1]By data providers, we refer to the subjects about whom the data is stored.

3. PURPOSE

In the various models proposed for privacy protection, the purpose for accessing a data element plays a major role in determining whether the access should be permitted or denied. The reason is that a privacy policy mainly concerns with which data object is used for what purpose(s). For example, a typical privacy policy such as "we will collect and use customer identifiable information for billing purposes and to anticipate and resolve problems with your service" states that the information can be accessed only for the purposes of billing, customer service, and possibly some analysis. It is an obvious consequence that purpose is a central concept in any privacy protecting access control model, and yet the concept of purpose has not been thoroughly investigated. In this section, we formally define the notion of purpose and discuss key related issues.

3.1 Definition of Purposes

In order to preserve the privacy of data providers, every access to any piece of information must be controlled by privacy policies to which data providers have agreed. A typical privacy policy for a data element includes purpose(s), retention, condition, and obligation, and it states that the particular data element can be accessed only for the specific purpose(s) on the specific condition. The retention indicates how long the data element can be retained, and the obligation designates the actions that must be followed after an access to the data element is allowed. The aspect that is most interesting to us in this paper is the purpose as the purpose directly dictates how accesses to data items should be controlled. P3P defines purpose as "the reason(s) for data collection and use" and specifies a set of purposes, including *current, admin, develop, contact, telemarketing* [18]. The support of purpose information must address two major requirements. The first requirement is that in common business environments purposes naturally have a hierarchy based on the principles of generalization and specialization. This requirement suggests arranging purposes according to some hierarchical organization which also simplifies the management of purposes. The second requirement is that a system often interacts with other autonomous systems. In order to resolve any semantic conflict during interactions, we adapt ideas from current work on ontologies. An ontology consists of a set of concepts together with relationships defined among these concepts. The concepts and their relationships are generally described using a formal language. In our work, we use an ontology for binding each purpose with a set of keywords that are semantically equivalent. The next definition formalizes the above notions.

Definition 1. (*Purpose and Purpose Ontology*) A purpose describes for what reasons data is collected or used and it is defined as a tuple ⟨Conceptual-name, Keyword-set⟩. A purpose ontology is defined as $\langle \mathcal{P}, \leq \rangle$, where:

- $\mathcal{P}$ is a set of purposes such that for any two distinct purposes in $\mathcal{P}$, $p_i = \langle \text{Conceptual-name}_i, \text{Keyword-set}_i \rangle$ and $p_j = \langle \text{Conceptual-name}_j, \text{Keyword-set}_j \rangle$, the following conditions hold:
  1. $\text{Conceptual-name}_i \neq \text{Conceptual-name}_j$
  2. $\text{Keyword-set}_i \cap \text{Keyword-set}_j = \emptyset$
- $\leq$ is a partial order defined over $\mathcal{P}$. Let p_i, $p_j \in \mathcal{P}$. We say that p_i is a specialization of p_j if $p_i \leq p_j$. □

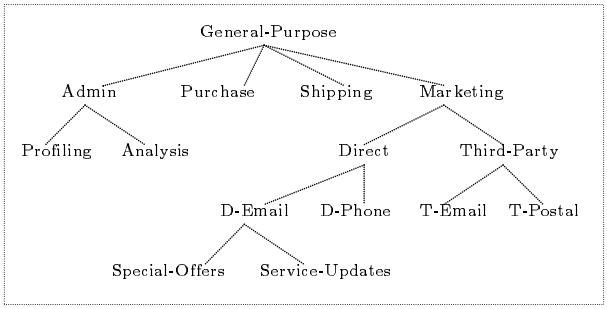


Figure 1: Purpose Tree

In this paper, we assume that the partial order is a tree[2] which we refer to as *Purpose Tree*. Figure 1 gives an example of purpose tree, where for simplicity we show only the conceptual names of purposes.

The following notations will be used through out this paper.

Notation 1. (*Root, Ancestors, and Descendants*) Let $\mathcal{PT}$ be a purpose tree and $\mathcal{P}$ be the set of purposes in $\mathcal{PT}$. Let p_i be a purpose in $\mathcal{PT}$.

1. Root($\mathcal{PT}$) is the root node of $\mathcal{PT}$, and it represents the most general purpose in $\mathcal{PT}$.
2. Ancestors(p_i) is the set of all nodes that are ancestors of p_i in $\mathcal{PT}$, including p_i itself.
3. Descendants(p_i) is the set of all nodes that are descendants of p_i in $\mathcal{PT}$, including p_i itself.

Example 1. Let $\mathcal{PT}$ be the purpose tree in Figure 1.

1. Root($\mathcal{PT}$) = {General-Purpose}
2. Ancestors(Analysis) = {Analysis, Admin, General-Purpose}
3. Descendants(Third-Party) = {Third-Party, T-Email, T-Postal}

Intuitively, an access to a specific data element is allowed if the purposes allowed by the privacy policies for the data include or imply the purpose for accessing the data. We refer to purposes associated with data and thus regulating data accesses as *Intended Purposes*, and to purposes for accessing data as *Access Purposes*. Intended purposes can be viewed as brief summaries of privacy policies for data, stating for which purposes data can be accessed. When an access to data is requested, the access purpose is checked against the intended purposes for the data.

Most privacy policies are positive in nature in that they selectively allow data access for a set of purposes. Thus, the absence of a particular purpose from the set of allowed purposes is interpreted as that data access for the purpose is not allowed. This concept is adopted in privacy policy languages such as P3P [18]. However, some privacy policies

[2]It is advantageous to organize purposes according to a DAG instead of a tree as it allows us to use a node to represent multiple purposes that have a similar meaning. For instance, D-Email and T-Email in Figure 1 could be represented in a single node Email in a DAG. However, this introduces ambiguous semantics as Email alone does not provide sufficient information; e.g., Email could be a specialization of either Direct or Third-Party. This subject of purpose management will be investigated in our future work.

may explicitly prohibit access to data for certain purposes. For example, suppose that in order to comply with COPPA [6], a company decides not to use any information about children of age under 13 for *Marketing*. This policy is negative in nature as it explicitly prohibits access to the data items belonging to minors for a particular purpose.

Our design of intended purposes supports both positive and negative privacy policies. An intended purpose consists of two components: *Allowed Intended Purposes* and *Prohibited Intended Purposes*. This structure provides greater flexibility to our access control model. Moreover, by using prohibited intended purposes, we can guarantee that data accesses for particular purposes are never allowed. Conflicts between the allowed intended purposes and the prohibited intended purposes for the same data element are resolved by applying the denial-takes-precedence policy where prohibited intended purposes override allowed intended purposes.

Definition 2. (*Intended Purpose*) Let $\mathcal{PT}$ be a purpose tree and $\mathcal{P}$ be the set of purposes in $\mathcal{PT}$. An intended purpose, IP, is a tuple $\langle$AIP, PIP$\rangle$, where AIP = {aip_1, ..., aip_n} is a set of allowed intended purposes and PIP = {pip_1, ..., pip_m} is a set of prohibited intended purposes. AIP and PIP satisfy the following two conditions:

1. (AIP $\subseteq \mathcal{P}$) $\wedge$ (PIP $\subseteq \mathcal{P}$)
2. AIP $\neq \emptyset$

Let AIP be a set of allowed intended purposes and PIP be a set of prohibited intended purposes, respectively. We define the set of intended purposes entailed by AIP and PIP as follows:

1. $\text{AIP}^* = \bigcup_{\text{aip}_j \in \text{AIP}} \text{Descendants}(\text{aip}_j)$
2. $\text{PIP}^* = (\bigcup_{\text{pip}_k \in \text{PIP}} \text{Ancestors}(\text{pip}_k)) \cup (\bigcup_{\text{pip}_k \in \text{PIP}} \text{Descendants}(\text{pip}_k))$ □

The condition "AIP $\neq \emptyset$" in the definition above ensures that every data element can be accessed for some purpose(s) unless it is explicitly prohibited by the corresponding PIP; we assume that data items without any allowed intended purpose should not be collected at all. Note that a complete block of data access can still be achieved by using PIP if needed.

Example 2. Suppose IP = $\langle${Admin, Direct}, {D-Email}$\rangle$ is defined over the purpose tree given in Figure 1.

1. AIP^* = Descendants(Admin) $\bigcup$ Descendants(Direct) = {Admin, Profiling, Analysis} $\bigcup$ {Direct, D-Email, D-Phone, D-Postal, Special-Offers, Service-Updates}
2. PIP^* = Descendants(D-Email) $\bigcup$ Ancestors(D-Email) = {D-Email, Special-Offers, Service-Updates} $\bigcup$ {D-Email, Direct, Marketing, General-Purpose}

It is important to note that the use of both AIP and PIP is not strictly necessary. In fact, IP = $\langle$AIP, PIP$\rangle$ can be always transformed into IP$'$ = $\langle$AIP$'$, $\emptyset\rangle$, where AIP$'$ = AIP^* − PIP^*. IP and IP$'$ are semantically equivalent. However, we decide to use both AIP and PIP in our model for the following reasons. First, as previously mentioned, some privacy policies are naturally positive whereas some are naturally negative. The support for both types of policies is valuable as it minimizes the possibility of implementation errors. Second, using PIP as exceptions, IP can be expressed in a more compact manner. For instance, suppose a privacy policy allows data access for any purpose except for *Third-Party* in Figure 1. This can be simply expressed as IP = $\langle${General-Purpose}, {Third-Party}$\rangle$ using both AIP and PIP while using AIP only, IP = $\langle${Admin, Purchase, Shipping, Direct}, $\emptyset\rangle$. Lastly, using PIP one can make sure that data access for particular purpose(s) is never allowed. This guarantee is often required for organizations who wish to keep their data management practice in compliance with privacy laws.

An *access purpose* is the purpose of a particular data access, which is determined or validated by the system when the data access is requested. How to determine or verify access purposes is not trivial, and this issue has not been thoroughly investigated in previously proposed models. We discuss this issue in detail in Section 3.3. We now formally define access purpose.

Definition 3. (*Access Purpose*) Let $\mathcal{PT}$ be a purpose tree. An access purpose, denoted by AP, is a purpose for accessing a data element, and it is a node in $\mathcal{PT}$. □

As already discussed, an access decision is made based on the relationship between the access purpose and the intended purposes of data. That is, an access is granted if the access purpose is entailed by the allowed intended purposes and not entailed by the prohibited intended purposes; in this case we say the access purpose is *compliant* with the intended purpose. The access is denied if any of these two conditions fails; we then say that the access purpose is *not compliant* with the intended purpose.

Definition 4. (*Access Purpose Compliance*) Let $\mathcal{PT}$ be a purpose tree. Let IP = $\langle$AIP, PIP$\rangle$ and AP be an intended purpose and an access purpose defined over $\mathcal{PT}$, respectively. AP is said to be compliant with IP according to $\mathcal{PT}$, denoted as AP $\Rightarrow_{\mathcal{PT}}$ IP, if and only if the following two conditions are satisfied:

1. AP $\notin \text{PIP}^*$
2. AP $\in \text{AIP}^*$ □

Example 3. Let $\mathcal{PT}$ be the purpose tree in Figure 1, and let IP and AP be an intended purpose and an access purpose defined based on $\mathcal{PT}$, respectively.

1. Suppose IP = $\langle${General-Purpose}, {Third-Party}$\rangle$. If AP = Marketing, then AP $\not\Rightarrow_{\mathcal{PT}}$ IP as Marketing $\in \text{PIP}^*$. However, if AP = Admin, then AP $\Rightarrow_{\mathcal{PT}}$ IP as Marketing $\notin \text{PIP}^*$ and Marketing $\in \text{AIP}^*$.
2. Suppose IP = $\langle${Admin, Purchase, Shipping}, {General-Purpose}$\rangle$. Then no AP defined over PT is compliant with IP.
3. Suppose IP = $\langle${General-Purpose}, $\emptyset\rangle$. Any AP defined over PT is compliant with IP.

3.2 Intended Purpose Management

Based on the purpose tree, an intended purpose is then specified for each data element according to the privacy policy on which the data provider has agreed. We assume that the organization has already established a set of comprehensive privacy policies which are compliant with existing privacy laws and that, as a part of the data collection process, data providers are informed of and agree to the privacy policies via some mechanism such as P3P [18]. Under this

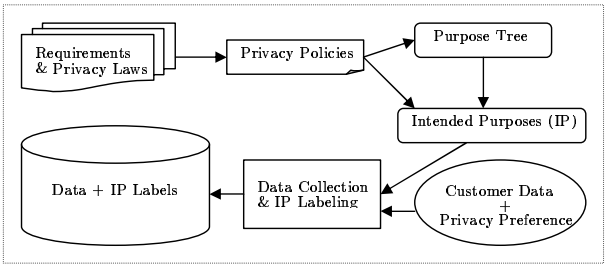


Figure 2: Intended Purpose Management Process

assumption, the privacy policy concerning each data element is predetermined; consequently, the intended purposes of most data items are predetermined.

We use the phrase "most data items" deliberately as in some exceptional cases the intended purpose for certain data can vary depending on individual data providers. For instance, enterprises who wish to fully comply with COPPA should have a different set of intended purposes concerning data collected from children under thirteen. Another example can be found in enterprises whose business activities span multiple nations. As different regions have different privacy requirements, such enterprises must carefully differentiate data providers from various regions and apply appropriate privacy policies (e.g., a set of intended purposes) to the collected data. The intended purposes can also vary due to additional options provided by enterprises. A common example of these options is the enrollment of a mailing-list. Whether or not the email address of a particular customer can be used for sending information about new services is dependent on the explicit decision of the customer. Note that the intended purposes can be further individualized upon the requests from data providers after data collection as well. The overall organization of the purpose management process is illustrated in Figure 2.

Example 4. Suppose a company has established the following privacy policies.

- We use your information for purchasing purposes, to provide services to you, and to inform you of services that may better meet your needs.
- We will not disclose your information to third parties who want to market products to you unless you allow us to do so.
- We do not use information of children under thirteen for any purpose other than providing requested services.
- The web server administrators may collect some data such as your IP address, referrer, and your web browser information. We do not make use of this information, but it may be used by system administrators to provide better service to you.

Table 1 illustrates the intended purposes for the data collected by the company, based on the privacy policies above and the purpose tree in Figure 1. Group 1 represents customers who are not children and have given consents for third-party marketing, and Group 2 represents customers who are not children and have not given consents for third-party marketing.

	Under 13	Group 1	Group 2
name	⟨{G},{A,M}⟩	⟨{G},∅⟩	⟨{G},{T}⟩
address	⟨{G},{A,M}⟩	⟨{G},∅⟩	⟨{G},{T}⟩
phone	⟨{G},{A,M}⟩	⟨{G},∅⟩	⟨{G},{T}⟩
order-history	⟨{G},{A,M}⟩	⟨{G},∅⟩	⟨{G},{T}⟩
web-log	⟨{A},{A,M}⟩	⟨{A},∅⟩	⟨{A},{T}⟩

G = General, A = Admin, P = Purchase, S = Shipping, M = Marketing, T = Third-party

Table 1: Predetermined Intended Purposes

3.3 Access Purpose Determination

An access purpose is the reason for accessing a data element, and it must be determined by the system when a data access is requested. How the system determines the access purpose of an access request is crucial as the access decision is made directly based on the access purpose. In this section we discuss some possible methods for determining access purposes.

First, the users can be required to state their access purpose(s) along with the requests for data access. Even though this method is simple and can be easily implemented, it requires complete trust on the users and the overall privacy that the system is able to provide entirely relies on the users' trustworthiness. Another possible method is to register each application or stored-procedure with an access purpose. As applications or stored-procedures have limited capabilities and can perform only specific tasks, it can be ensured that data users use them to carry out only certain actions with the associated access purpose. This method, however, cannot be used for complex stored-procedures or applications as they may access various data for multiple purposes. Lastly, the access purposes can be dynamically determined, based on the current context of the system. For example, suppose an employee in the shipping department is requesting to access the address of a customer by using a particular application in a normal business hour. From this context (i.e., the job function, the nature of data to be accessed, the application identification, and the time of the request), the system can reasonably infer that the purpose of the data access must be shipping.

In this paper, we assume the first method where the users are required to explicitly state their access purpose(s) when they try to access data. That is, the users provide an access purpose for each query they issue. To make our discussion more concrete, we extend SQL queries as well as update commands by adding an additional clause for access purposes. For instance, a simple select statement "SELECT name FROM customer" is extended to a form of "SELECT name FROM customer FOR marketing". Our proposed SQL extensions are provided in the Appendix.

We are currently extending this simple idea by adding a validation process which verifies the stated access purposes. When a query with an access purpose is submitted, the system first tries to validate the access purpose by verifying them with the data user's attributes such as currently activated role(s), job position, and location. If the validation fails, the query is rejected without being further processed. Here we assume that after data users are authenticated, their user identifications and their attributes are available to the access control mechanism. This extension will be further developed in our future work. Figure 3 illustrates the overall query execution process.

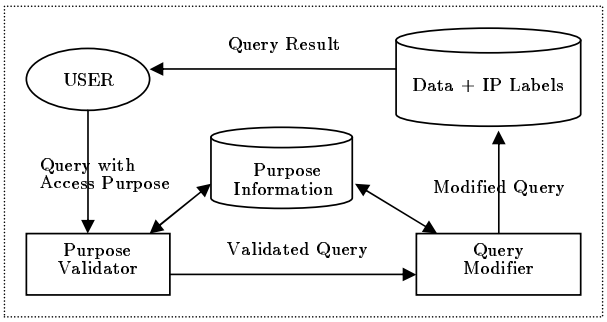


Figure 3: Query Process Steps

4. DATA LABELING MODEL

In order to build an access control model based on the notion purpose, we must consider a specific data model and based on this model devise a proper labeling scheme. A major question here is how intended purposes are associated with data. More specifically, we have to determine at what level of granularity data will be associated with intended purposes. For instance, consider the relational data model. Under such a model, intended purpose can be assigned to every relation, to every tuple in every relation, to every attribute in every relation, or to every data element in every relation.

Data providers (e.g., customers) are usually reluctant to allow any use of their information unless it is absolutely necessary. At the same time, data users (e.g., enterprises) want to make use of the collected data for the necessary tasks as well as other tasks such as analysis and marketing. Consequently, some form of negotiating process may occur between these two parties through opt-in/opt-out procedures. Note that the comfort level of privacy can considerably vary from individual to individual. Consider a fictional data category *purchase history*, which consists of customer name, financial-info, product, and purchase-date. As this category includes sensitive information such as financial-info, many customers would not want to allow any use of the information in this category. However, information such as name, product, and purchase-date can be very valuable for enterprises as they can use such information for analyzing their sale patterns or profiling customers. It is obvious that in order to make the best use of data while at the same time ensure that data providers feel comfortable, the granularity of the data labeling model must be fine. Thus, the labeling model should allow the assignments of intended purposes with data at the most fine-grained level. That is, we should be able to assign an intended purpose to each data element in every tuple; e.g., for each attribute and for each data provider (see Table 2)[3].

However, this most fine-grained approach is not always necessary. For instance, some data naturally have a hierarchical structure. Suppose that the addresses of customers are stored in a relation that consists of street, city, state, and postal-code. Typically, a customer allows or prohibits access

[3]The Tables 2-6 only represent a logical view and are intended to be used only for illustration. The actual implementation and organization of purpose labeling will be discussed later in the paper.

c_id	c_id_ip	name	name_ip	income	income_ip
1001	⟨{G},∅⟩	John	⟨{G},{M}⟩	110,000	⟨{A},{M}⟩
1002	⟨{G},∅⟩	Paul	⟨{G},∅⟩	56,000	⟨{G},∅⟩
1003	⟨{G},∅⟩	Jack	⟨{G},∅⟩	48,000	⟨{G},{T}⟩

G = General, A = Admin, M = Marketing, T = Third-party

Table 2: Customer Table

c_id	street	city	state	zip_code	addr_ip
1001	32 Oval Dr	Lafayette	IN	47907	⟨{G},{A,M}⟩
1002	433 State Rd	Chicago	IL	46464	⟨{G},∅⟩
1003	199 First Ave	Boston	CA	02139	⟨{G},{T}⟩

G = General, A = Admin, M = Marketing, T = Third-party

Table 3: Address Table

to the entire address, not to the individual sub-elements. Thus, it is not necessary to associate each data element with an intended purpose because labeling in the element-level granularity would result in storing an identical intended purpose for every data element redundantly. However, intended purpose for the address can vary depending on each individual. To address these concerns, the labeling model should allow the assignment of intended purpose to each tuple of a relation (see Table 3).

Another case we should consider is that there exists some information for which corresponding privacy policies are mandated by enterprises or by laws; i.e., data providers do not have a choice to opt-out from the required intended purposes. An example is the *Order* information in Table 4. As such information must be accessed to perform necessary tasks, enterprises can choose not to give customers any option to change the privacy policies governing this information. In such cases, the data elements in each column in the relation have the identical intended purpose. Thus, in order to avoid any redundancy, intended purposes should be assigned to each attribute of a relation using a *privacy policy table* (see Table 5). It is also possible that the intended purposes of every attribute in a relation be identical. Such cases occur when information in a relation is meaningful as a whole tuple, but individual elements or tuples do not have any usefulness. The *Access-Log* table in Table 6 is one such relation. In this case, the intended purposes are assigned to the entire relation by using a single entry in the privacy policy table (see Table 5).

Now we define our labeling model more formally.

Definition 5. (*Relation, Attribute, Tuple, and Element*) As in the standard relational model, data are stored in relations. A relation is characterized by the following two components.

1. A state-invariant relation scheme $\mathcal{R}(A_1, \ldots, A_n)$, where $\mathcal{R}$ is the name of the relation and each A_i is an attribute over some domain D_i. Attributes($\mathcal{R}$) denotes the set of names of attributes in $\mathcal{R}$.
2. A state-dependent relation instance r_i[4] over $\mathcal{R}$ composed of distinct tuples of the form $(a_1, \ldots, a_n)$, where

[4]We assume that relation names are used as arguments of relational algebraic expressions.

or_id	c_id	product	credit_info	date	status
101	1001	P303	V3434-343-2222	10/23/03	shipped
102	1002	P887	V5675-374-5892	07/20/04	packaged
103	1003	S99-6	M6584-677-4911	08/22/04	ordered

Table 4: Order Table

table_name	column_name	ip
order	product	⟨{A, P, S}, ∅⟩
order	credit_info	⟨{P}, {M}⟩
order	date	⟨{A, P, S}, {M}⟩
order	status	⟨{A, P, S}, ∅⟩
access_log	ALL	⟨{A, P}, ∅⟩

A = Admin, P = Purchase, S = Shipping, M = Marketing

Table 5: Privacy-Policy Table

client_ip	date	time	requested_url
4.33.163.99	15/08/04	18:35:22	/sci-fi/books/index.html
218.232.444.33	15/08/04	19:35:53	/home.html
63.344.343.75	15/08/04	19:36:02	/kids/music/index.html

Table 6: Access-Log Table

each element a_i is a value in domain D_i. □

Definition 6. (*Intended Purpose Labeling*) Let $\mathcal{PT}$ be a purpose tree and $\mathcal{P}$ be the set of purposes in $\mathcal{PT}$. Let $\mathcal{IP}$ be a set of all possible intended purposes defined over $\mathcal{P}$ and $\mathcal{R}(A_1, \ldots, A_n)$ be a relation. In our data labeling model, intended purposes are associated with $\mathcal{R}$ according to one of the following methods.

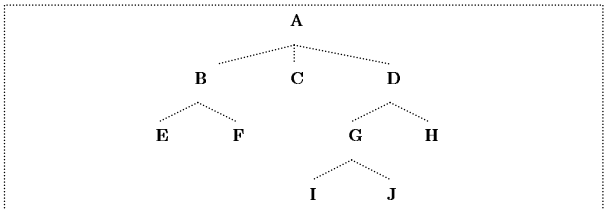
1. (*Relation-based*) A relation-based labeling is a pair $\langle \mathcal{R}, ip \rangle$, where $ip \in \mathcal{IP}$. Access to any data element in instances of $\mathcal{R}$ is governed by ip.
2. (*Attribute-based*) An attribute-based labeling is a set $\{\langle A_i, ip_i \rangle \mid A_i \in Attributes(\mathcal{R}) \wedge ip_i \in \mathcal{IP}\}$. Access to data element a_i in any instance of $\mathcal{R}$ is governed by ip_i.
3. (*Tuple-based*) A tuple-based labeling is a relation scheme $Rtl(A_1, \ldots, A_n, \ell)$, where ℓ is a column having $\mathcal{IP}$ for its domain, such that $\mathcal{R} = \prod_{A1,\ldots,An}(Rtl)$. Access to any data element in the jth tuple in any instance of $\mathcal{R}$ is governed by ℓ_j.
4. (*Element-based*) An element-based labeling is a relation scheme $Rel(A_1, \ell_1, \ldots, A_n, \ell_n)$, where ℓ_i ($i = 1, \ldots, n$) is a column having $\mathcal{IP}$ for its domain, such that $\mathcal{R} = \prod_{A1,\ldots,An}(Rel)$. Access to data element a_i in any instance of $\mathcal{R}$ is governed by ℓ_i. □

The first two types of labeling are intensional labeling schemes as they are defined at schema level. On the other hand, the third and fourth types are extensional labeling schemes as they are associated with data elements inside relation extensions.

The element-based labeling scheme is illustrated by Table 2, where each data element is labeled with an intended purpose. Table 3 is an example of the tuple-based labeling scheme, and here intended purposes are associated with each tuple. Tables 4, 6, and 5 illustrate the relation- and attribute-based labeling schemes. Note that these tables only represent a logical view. We discuss actual implementation strategies in the following section.

5. IMPLEMENTATION

In this section we discuss two important implementation issues. The first is related to approaches for storing purpose information and for recording which purposes are associated with which data elements. The other issue is related to query modification techniques supporting data filtering with respect to the notion of purpose.



(i) Purpose Tree

p_id	p_name	parent	code	aip_code	pip_code
1	A	-	0x200	0x3FF	0x3FF
2	B	1	0x100	0x130	0x330
3	C	1	0x080	0x080	0x280
4	D	1	0x040	0x04F	0x24F
5	E	2	0x020	0x020	0x320
6	F	2	0x010	0x010	0x310
7	G	4	0x008	0x00B	0x24B
8	H	4	0x004	0x004	0x244
9	I	7	0x002	0x002	0x24A
10	J	7	0x001	0x001	0x249

(ii) pt_table

Figure 4: Purpose Tree Storage

5.1 Privacy Metadata Storage

For both storage and performance efficiency, purposes are encoded as bit strings. Consider the purpose tree in Figure 4(i). This purpose tree is encoded into a relation *pt_table* as shown in Figure 4(ii). The first column *p_id* represents the identification number of each purpose node, which is determined according to the breadth-first search order of the tree. The second column *p_name* represents the name of each purpose node, and the third column *parent* is used to capture the hierarchical relationships among the purpose nodes. The column *code* is the binary encoding of each purpose. For instance, in Figure 4 the purpose A is encoded as binary number '1000000000', which is equivalent to '0x200' in hexadecimal representation, while the purpose D is encoded as '0001000000', which is '0x040' in hexadecimal form. Note that a binary representation provides significant advantages in that it is very efficient in terms of storage and computation.

The last two columns *aip_code* and *pip_code* are precalculated encodings of purpose entailments. As described in Section 3, when a purpose p_i is used as an AIP, it entails that every descendant of p_i, including p_i itself, is allowed. For instance, the purpose B in Figure 4(i) used as an AIP entails that access is allowed for the purpose of B as well as E and F. Thus, the aip_code of B contains the entailed set of B in that the value is the sum of the codes of B, E and F. Similarly, when a purpose p_i is used as a PIP, it entails that every descendant and every ancestor of p_i, including p_i itself, is prohibited. For instance, the purpose B in Figure 4(i) used as a PIP entails that access is prohibited for the purpose of B as well as E, F, and A; thus, the pip_code of B is the sum of the codes of B, E, F, and A. Note that the last three columns of the *pt_table* can be automatically generated based on the first three columns using a simple procedure.

The other type of metadata to be stored is the intended purposes for the actual data. As described in Section 4, data elements are associated with intended purposes according to one of the intended purpose labeling schemes. When using

the element-based labeling scheme, a table with n columns is extended to *(n + 2n)* columns[5]; n data columns plus *2n* columns for AIP and PIP. When using the tuple-based labeling scheme, a table with n columns is extended to *(n + 2)* columns; n data columns plus *2* columns for AIP and PIP. While the element- and tuple-based labeling schemes require extensions to data tables, the attribute-based labeling scheme for a table with n columns requires n entries in the privacy policy table as shown in Table 5. Similarly, the relation-based labeling scheme for a table requires a single entry in the privacy policy table.

As every data element is associated with an intended purpose, the size of the metadata increases as the size of actual data grows. Especially for element-based labeling scheme, the storage overhead cost can be very high. However, this is not a practical issue as the number of data items which require the element-based labeling scheme is relatively small in practice. In fact, typical privacy policies would specify intended purposes for most data items without providing any option for opt-in or opt-out. As the intended purposes of these data items are not changeable and cannot be individualized, the attribute- or relation-based labeling schemes should be used for these data items. Therefore, the element- or tuple-based labeling schemes will be used only for a few data items with opt-in or opt-out options.

Intended purposes are encoded using the purpose encodings in the *pt_table*. As intended purposes have their entailments, purposes will be encoded using values from the *aip_code* or the *pip_code* instead of using values in the *code*. Also, purposes can be combined by performing bitwise OR operations on the encodings of the purposes. Consider, for example, the intended purpose of a data element is $\langle\{B, C\}, \{G\}\rangle$ with respect to the purpose tree in Figure 4(i). Then the AIP of the data element is encoded as '0x1B0', which is the result of (0x130 | 0x080), where | is bitwise OR operator. The PIP is simply represented as '0x24B', which is the pip_code of G. Note that using stored procedures and GUI tools, the management of intended purposes can be easily carried out.

With this encoding method, the purpose compliance can be efficiently checked. Recall that an access purpose is compliant with an intended purpose if and only if the access purpose is not prohibited by PIP and it is allowed by AIP. Thus, the purpose compliance check can be done with two bitwise AND operations as follows. Given the encodings of an access purpose[6], AIP and PIP, say ap_code, aip_code and pip_code respectively, the access purpose is compliant with the intended purpose if and only if (ap_code & pip_code) $= 0 \wedge$ (ap_code & aip_code) $\neq 0$, where & is bitwise AND operator and $\wedge$ is logical AND operator.

5.2 Access Control Using Query Modification

Privacy-preserving access control mechanisms must ensure that a query result contains only the data items that are allowed for the access purpose of the query. In other words, the system must check the intended purpose of each data element accessed by the query and filter out its value

[5]A different method is to store intended purposes in separate tables. However, storing data and intended purpose in separate tables will cause additional overhead introduced by join operations.

[6]Access purposes are represented using the values in the *code* of the *pt_table*.

```
Comp_Check (Number ap, Number aip, Number pip)
Returns Boolean
1.   if (ap & pip) ≠ 0 then
2.     return False;
3.   else if (ap & aip) = 0 then
4.     return False;
5.   end if;
6.   return True;

Modifying_Query (Query Q)
Returns a modified privacy-preserving query Q'
1.   Let R1, ..., Rn be the relations referenced by Q
2.   Let P be the predicates in WHERE clause of Q
3.   Let a1, ..., am be the attributes referenced in both
     the projection list and P
4.   Let AP be the access purpose encoding of Q
5.
6.   for each Ri where i = 1, ..., n do
7.     if (Ri is relation-based labeling AND
         Comp_Check (AP, Ri.aip, Ri.pip) = False) then
8.       return ILLEGAL-QUERY;
9.     else if Ri is attribute-based labeling then
10.      for each aj which belongs to Ri do
11.        if Comp_Check (AP, aj.aip, aj.pip) = False then
12.          return ILLEGAL-QUERY;
13.        end if;
14.      end for;
15.    else if Ri is tuple-based labeling then
16.      add ' AND Comp_Check (AP, Ri_aip, Ri_pip)' to P ;
17.    else if Ri is element-based labeling then
18.      for each aj which belongs to Ri do
19.        add ' AND Comp_Check (AP, aj_aip, aj_pip)' to P;
20.      end for;
21.    else // Ri is a relation without labeling
22.      do nothing;
23.    end if;
24.  end for;
25.
26.  return Q with modified P;
```

Figure 5: Query Modification Algorithm

if the access purpose is not compliant with the intended purpose of the data element. In our implementation, this fine-grained access control is achieved using query modification [16]. Note that query modification provides powerful and flexible controls without requiring any alteration in underlying mechanisms.

Our query modification algorithm is outlined in Figure 5. Note that this method is invoked only if the access purpose of the query is verified to be acceptable by the validate function, as described in Section 3. If the access purpose is not acceptable, then the query is rejected without further being processed.

In Lines 7 and 9 the compliance checks for relations with the relation- or attribute-based labeling schemes are executed statically by the query modification method. On the other hand, the compliance checks for relations with the tuple- or element-based labeling schemes are performed during query processing by the predicates which are added by the query modification algorithm (Lines 15 and 17).

The query modification algorithm checks both the attributes referenced in the projection list and the attributes referenced in predicates (Line 3). As the attributes in the projection list determine what data items will be included in the result relation of a query, it may seem enough to enforce privacy policy based only on the attributes in the projection list. However, the result of a query also depends on the predicates, and not enforcing privacy constraints on the predicates may introduce inference channels. For example,

Query	Modified Query
Select name, phone **From** customer **For** Marketing	**Select** name, phone **From** customer **Where** Comp_Check('0x200', name_aip, name_pip) **and** Comp_Check('0x200', phone_aip, phone_pip)
Select name, city **From** customer **as** C, address **as** A **Where** C.c_id = A.c_id **For** Shipping	**Select** name, city **From** customer **as** C, address **as** A **Where** C.c_id = A.c_id **and** Comp_Check('0x400', Address_aip, Address_pip) **and** Comp_Check('0x400', A.name_aip, A.name_pip) **and** Comp_Check('0x400', A.c_id_aip, A.c_id_pip)
Select product **From** order **Where** c_id = 1101 **For** Profiliing	**Select** product **From** order **Where** c_id = 1101

Table 7: Query Modification Examples

consider the following query: "**Select** name **From** customer **Where** income > 100000 **For** Third-Party". Suppose that according to the established privacy policies, *name* can be accessed for the purpose of *Third-Party*, but *income* is prohibited for this purpose. If the privacy constraint is not enforced on the predicates, this query will return a record containing the names of customers whose income is greater than 100,000. This is highly undesirable as this result implicitly conveys information about the customers' income. Note that if the privacy policy is enforced at the predicate level, such inference channels cannot be created.

Notice that the provided algorithm filters out a tuple if any of its elements that are accessed is prohibited with respect to the given access purpose. For instance, consider the following query: "**Select** name, phone **From** customer **For** Marketing". Suppose there is a customer record of which the *name* is allowed for marketing, but the *phone* is prohibited for this purpose. Then our algorithm excludes the record from the query result. We note that in the environments where partially incomplete information is acceptable, the query modification algorithm can be easily modified to mask prohibited values with null values using the case expression in SQL.

Example 5. Table 7 illustrates how queries are modified by our algorithm. Tables 2-6 are used for this example. Note that the purpose encodings of *Marketing* and *Shipping* are assumed to be '0x200' and '0x400', respectively.

6. EXPERIMENTAL EVALUATION

The main goal of our experiment is to investigate performance and storage overheads of our approach. As the relation- and attribute-based labeling schemes do not add significant runtime overheads, we mainly focus on the tuple- and element-based labeling schemes. We consider the impact of the purpose hierarchy and compare the performance overheads of different labeling schemes. We also examine the response times of queries, varying the numbers of attributes accessed. Lastly, we test the scalability of our approach by experimenting with relations of different cardinalities.

6.1 Experimental Setup

The experiments were performed on a 2.66 GHz Intel machine with 1 GB of memory. The operating system on the machine was Microsoft Windows XP Professional Edition, and Oracle Database 10g Enterprise Edition Release 1 was used as our DBMS.

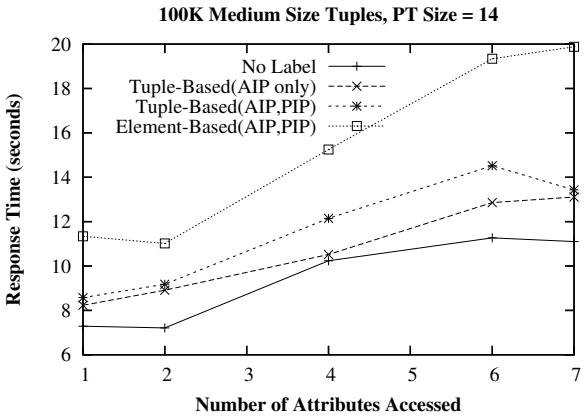


Figure 6: Labeling Scheme and Performance

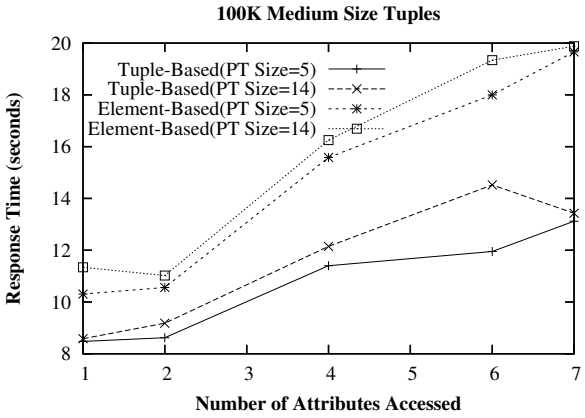


Figure 7: Purpose Size and Performance

For the experiments, we generated synthetic datasets, which were simpler versions of the Wisconsin Benchmark [5]. Then we extended each relation by adding intended purpose label columns, according to our labeling schemes described in Section 4. In order to precisely compare the overheads of different intended purpose labeling schemes, we set the selectivity of all data elements to 100 percent; i.e., all intended purpose labels are set to allow access for every access purpose.

After creating all necessary relations, we measured the response times[7] of various queries and modified versions[8] of those queries. We ran each query 10 times, flushing the buffer cache and the shared pool before each run.

6.2 Impact of Labeling Schemes and Purpose Hierarchy

Figure 6 shows the response times of queries against relations with various labeling schemes. As shown, the response time increases as the granularity of labeling scheme becomes

[7]To measure the response time of a query, we measured the time to retrieve the selected tuples into a relation; thus, the reported response times here include the time for inserting the selected tuples, in addition to the time for retrieving the tuples.

[8]As we had not implemented the query modification algorithm, each query was modified manually before the experiment. Our future work includes a full implementation of the query modification method in a public domain database management system.

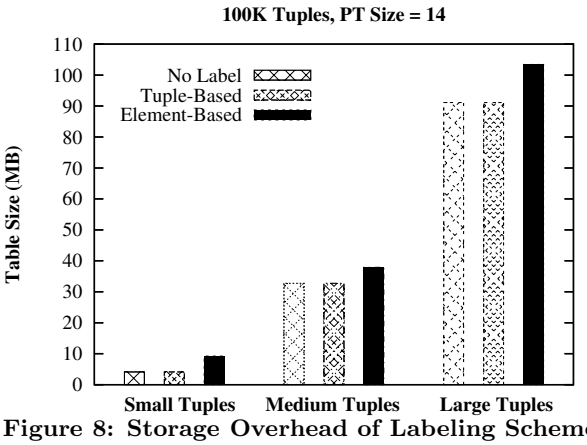


Figure 8: Storage Overhead of Labeling Scheme

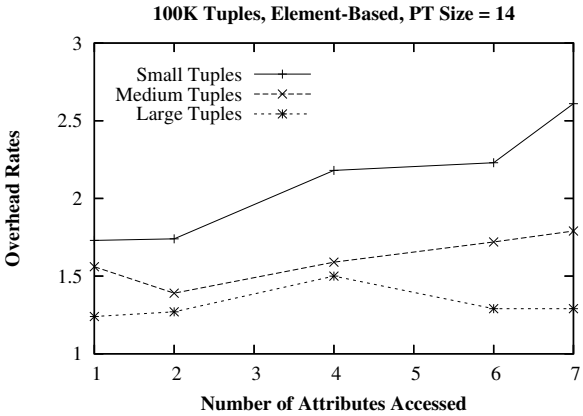


Figure 9: Storage Overhead and Performance

finer. This increase is indeed expected as more purpose-compliance checks are needed for finer labeling schemes. For the element-based labeling scheme, the number of attributes accessed by queries is also a major factor. As the element-based labeling scheme requires a compliance check for every element a query is accessing, the overhead for compliance checks becomes more significant as the number of accessed attributes increases. However, as the tuple-based labeling scheme requires only one purpose-compliance check for each tuple, the number of accessed attributes does not have any impact on the tuple-based labeling scheme. Figure 6 also shows that the use of both AIP and PIP does not introduce much overhead, compared to the case where only AIP is used. This is a reasonable result as using AIP and PIP requires only one additional bitwise-AND operation for a compliance check.

Figure 7 shows the results of our experiments with two different sizes of purpose trees. The first purpose tree has five nodes with the height of two, and it requires five bits to encode all possible intended purposes. The second purpose tree has 14 nodes with the height of five, requiring 14 bits for all possible intended purpose encodings. As the result shows, the size of purpose tree does not make any substantial difference in either the tuple- or element-based labeling scheme. The reason for this indifference is that bitwise-AND operations are very efficient regardless of the length of encodings.

6.3 Storage Overhead vs. Performance Overhead

In this section, we consider the storage overhead introduced by our intended purpose labeling schemes. In fact, the storage overhead of labeling schemes can be easily calculated as follows. Let r_c be the cardinality of the relation $\mathcal{R}$, t_s be the size of a tuple and n_c be the number of columns in $\mathcal{R}$. Also let p_s be the size of the purpose tree, and ℓ_s be the size of an intended purpose label; $\ell_s = (p_s \ / \ 8)$ in bytes. Then the size of the unextended relation is $(r_c \times t_s)$, and assuming the element-base labeling scheme, the size of the extended relation $\mathcal{R}_{ext}$ becomes $r_c \times (t_s + 2 \times n_c \times l_s)$. We remind readers that the element-based labeling scheme extends a relation with n columns to a relation with with $(n + 2n)$ columns; n data columns plus $2n$ columns for AIP and PIP. Thus, the storage overhead rate of the element labeling scheme becomes $1 - [Size(\mathcal{R}_{ext}) \ / \ Size(\mathcal{R})] = (2 \times n_c \times l_s) \ / \ t_s$. This clearly suggests that if the original relation contains large data elements (which we believe is a common case for databases storing information about individuals), the storage overhead of our labeling scheme becomes negligible. Figure 8 shows the storage overheads of the element-based labeling scheme with relations of three different tuple sizes; small, medium and large. Note that all three relations have the same cardinality of 100K.

The storage overhead has an effect on the performance of modified queries as well. Figure 9 shows performance overhead rates of modified queries for relations with three different tuples sizes. Here, the overhead rate is measured as [(*response time of the modified query*) / (*response time of the unmodified query*)]. In particular, the table with small-sized tuple is doubled in its size by intended purpose labeling, and the performance overhead on this extended table becomes very large. However, this is an extreme case as typical private data would be much larger in its size than the size of an intended purpose label.

6.4 Scalability

As our method filters out prohibited values by performing a purpose-compliance check for each tuple in case of the tuple-based labeling scheme or for each element in case of the element-based labeling scheme, the cardinality of relations can significantly impact performance. For instance, querying a relation with one million tuples will require at least one million compliance checks. Even though our implementation of the compliance check is merely two bitwise-AND operations, the runtime compliance checks can be a heavy burden for large databases. Figure 10 shows the performance overhead rates of the element-based labeling scheme for relations with various cardinalities. As expected, the performance becomes poorer as the cardinality of relations increases. However, this problem can be easily addressed by using function-based indexes [13]. The function-based index allows creating indexes on a function by pre-computing the given function. Thus, we can pre-build a function-based index for each possible access purpose and use these indexes when queries are executed. Figure 11 displays the results of our experiment with two indexes; a bitmap index and a B+ tree index. Notice that as the compliance check is always evaluated to either true or false (0 or 1), the bitmap index performs slightly better than B+ tree index. Nonetheless, a huge performance improvement is gained when either type of index is used. This shows that using indexes, our ap-

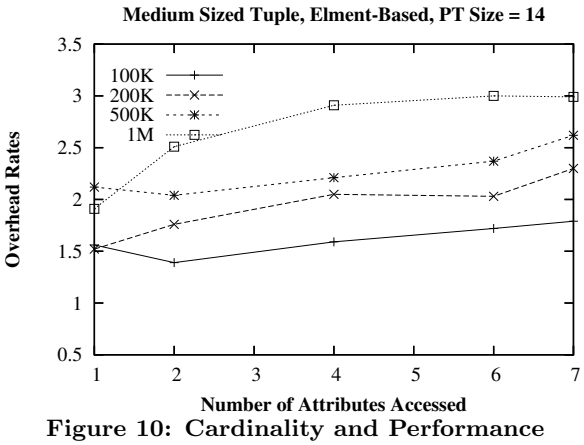


Figure 10: Cardinality and Performance

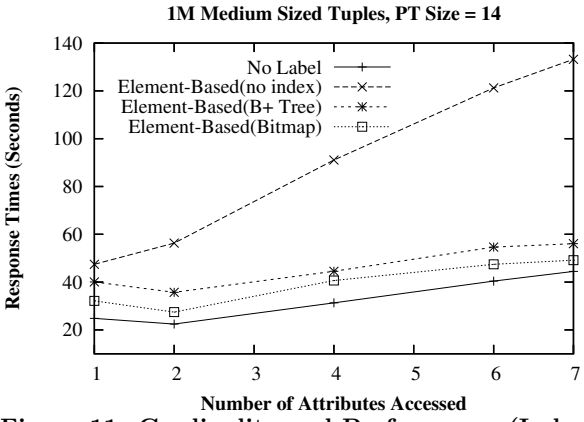


Figure 11: Cardinality and Performance (Index)

proach introduces very minimal performance overhead and is highly scalable.

7. CONCLUSION AND FUTURE WORK

In this paper, we proposed an access control model for privacy protection based on the notion of purpose. We discussed a comprehensive definition of purpose and introduced the concepts of intended purpose and access purpose. For data labeling, we presented four different labeling schemes, each providing a different granularity. We also discussed various implementation issues and suggested a method based on query modification. Through our experiments, we showed that our method introduces very minimal overheads in both storage and performance and is highly scalable.

Our proposed model provides a comprehensive framework for privacy preserving access control systems, but much more work still remains to be done. Our future work includes devising a high level language for purpose-oriented privacy policy which can be used to automatically manage the intended purposes of data. Compatibility issues with P3P will also be investigated. We also plan to extend our model to cope with other elements of privacy such as obligations and complex conditions. In order to achieve this, we will introduce event-based privacy management, which makes use of trigger mechanisms. We have also observed a need for specialized benchmarks for private data management systems. We also plan to investigate techniques according to which our logical proposed extensions to SQL could be supported without requiring actual extensions to the DBMS internals. Last but not least, we will explore the notion of sticky-policy paradigm [10]. The sticky-policy paradigm requires that the policy under which data have been collected governs the use of these data at all times. This is a challenging problem, but we believe that this is a vital element of privacy protection.

8. REFERENCES

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *The 28th International Conference on Very Large Databases (VLDB)*, 2002.

[2] P. Ashley, C. S. Powers, and M. Schunter. Privacy promises, access control, and privacy management. In *Third International Symposium on Electronic Commerce*, 2002.

[3] D. E. Bell and L. J. LaPadula. Secure computer systems: mathematical foundations and model. Technical report, MITRE Corporation, 1974.

[4] E. Bertino, S. Jajodia, and P. Samarati. Database security: Research and practice. In *Information Systems*, 1996.

[5] D. Bitton, D. J. DeWitt, and C. Turbyfill. Benchmarking database systems: a systematic approach. In *Ninth International Conference on Very Large Data Bases*, Nov. 1983.

[6] F. T. Commision. Children's online privacy protection act of 1998. Available at www.cdt.org/legislation/105th/privacy/coppa.html.

[7] F. T. Commission. Privacy online: Fair information practices in the electronic marketplace: A report to congress. Technical report, May 2000. Available at www.ftc.gov/reports/privacy2000/privacy2000.pdf.

[8] D. Denning, T. Lunt, R. Schell, W. Shockley, and M. Heckman. The seaview security model. In *The IEEE Symposium on Research in Security and Privacy*, 1998.

[9] IBM. *The Enterprise Privacy Authorization Language (EPAL)*. Available at www.zurich.ibm.com/security/enterprise-privacy/epal.

[10] G. Karjoth, M. Schunter, and M. Waidner. Platform for enterprise privacy practice: Privacy-enabled management of customer data. In *The 2nd Workshop on Privacy Enhancing Technologies (PET 2002)*, Apr. 2002.

[11] A. Kobsa. Personalized hypermedia and international privacy. *Communications of the ACM*, 2000.

[12] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. DeWitt. Disclosure in hippocratic databases. In *The 30th International Conference on Very Large Databases (VLDB)*, Aug. 2004.

[13] Oracle. *The Oracle Database SQL References*, Dec. 2003. Availabe at www.oracle.com.

[14] R. Sandhu and F. Chen. The multilevel relational data model. In *ACM Transaction on Information and System Security*, 1998.

[15] R. Sandhu and S. Jajodia. Toward a multilevel secure relational data model. In *ACM International Conference on Management of Data (SIGMOD)*, 1991.

[16] M. Stonebraker and E. Wong. Access control in a relational data base management system by query modification. In *ACM CSC-ER Proceedings of the 1974 Annual Conference*, Jan. 1974.

[17] World Wide Web Consortium (W3C). *A P3P Preference Exchange Language 1.0 (APPEL 1.0).* Available at www.w3.org/TR/P3P-preferences.

[18] World Wide Web Consortium (W3C). *Platform for Privacy Preferences (P3P).* Available at www.w3.org/P3P.

APPENDIX

A. SQL EXTENSIONS

A.1 Data Definition Language(DDL)

1. (***Table Creation***) The purpose labeling specification will be stored in the system catalog and the provided purposes will be used as default values.

   Create table-name {
   column-name1 data-type,
   column-name1 data-type,
   ...
   } [With *Labeling*]

   where *Labeling* is one of the followings:
   1. EBL(purpose1, purpose2, ...) : Element-based
   2. TBL(purpose) : Tuple-based
   3. ABL(purpose1, purpose2, ...) : Attribute-based
   4. RBL(purpose) : Relational-based

2. (***Column Addition***) The table must be element- or attribute-based labeling. If not provided a default purpose (e.g., none-allowed) will be used.

   Alter Table table-name
   Add column-name data-type [**With** purpose]

3. (***Column Removal***) No change is needed.

   Alter Table table-name
   Drop Column column-name

A.2 Data Manipulation Language(DML)

1. (***Query***) If **For** clause is not provided, the most general purpose (i.e., the root of the purpose tree) will be used.

   Select column-names
   From table-names
   Where column-name = some-value
   [**For** purpose]

2. (***Insertion***) The table must be tuple- or element based labeling. If **With** clause is not provided, the default purpose, specified at the creation of table, will be used.

   Insert into table-name
   Values (v1, v2, ...)
   [**With** (purpose1, purpose2, ...]

3. (***Deletion***) If **For** clause is not provided, the most general purpose (i.e., the root of the purpose tree) will be used.

   Delete from table-name
   Where column-name = some-value
   [**For** purpose]

4. (***Update***) If **For** clause is not provided, the most general purpose (i.e., the root of the purpose tree) will be used.

   Update table-names
   Set column-name = new-value
   Where column-name = some-value
   [**For** purpose]

A.3 Purpose Management Language(PML)

1. (***Purpose Creation***) The root of the purpose tree (e.g., General-Purpose) is initially created by the system.

   Create Purpose purpose-name
   Parent purpose-name

2. (***Purpose Deletion***) If the purpose is not a leaf, the descendants of the purpose will be deleted as well.

   Delete Purpose purpose-name

3. (***Intended Purpose View***) If the table is element- or tuple-based labeling, both column-name and value must be provided. For attribute-based labeling, only column-name is required. For relation-based labeling, none is required.

   View Purpose table-name
   [column-name] [= value]

4. (***Intended Purpose Update***) If the table is element- or tuple based labeling, both column-name$_1$ and **Where** clause must be provided. For attribute-based labeling, only column-name$_1$ is required. For relation-based labeling, none is required.

   Update table-names
   Set Purpose [column-name$_1$ =] new-Purpose
   [**Where** column-name$_2$ = some-value]