Carnival
An Application Framework for Enforcement of Privacy Policies

Ragni R. Arnesen, Jerker Danielsson, and Bjørn Nordlund

***Abstract*—This paper presents Carnival, a framework providing privacy access control and audit functionality to application developers. Carnival enforces privacy policies that regulate access based on the action requested, the identity and/or roles of the requesting user, the purpose of the access, the identity and preferences of the data subject associated with the data, and the type of personal data to be accessed. A description of our implementation of Carnival in Java, and an outline of how to develop and deploy applications using this framework, is provided.**

***Index Terms*—Access control, Privacy, Privacy policy enforcement.**

I. Introduction

The presence of extensive data collection and processing capabilities threatens the privacy of individuals and organizations. However, the inherent privacy intruding effect of these collection and processing practices can be substantially reduced.

Data collectors should analyze their current collection practices and evaluate the types and amount of data collected, whether the collection is "needed and worth it" and whether pseudonymized data or less granular data is sufficient for the purposes of the data collection. (See e.g. Hansen and Pfitzmann [11] for a definition of pseudonymity.) Furthermore, under some circumstances it is possible to let the data subject, i.e. the person whose identity is, or may be, connected to the data, remain in control over her personal data by letting her control the transformation between pseudonym and identifier. That is, the data subject controls the keys that unlock the pseudonyms.

However, there are circumstances where processing of identifiable personal data is both useful and necessary. For example, medical data must be collected and processed within the hospital sector, and banks need personal data to evaluate customers' credit. In other cases, processing of personal data (possibly pseudonymized) is not strictly necessary, but may be of benefit to both data collector and data subject. An example of such a case is the possibility for a data collector to customize offers to the data subject based on her interests, history, and current context, e.g. location.

In any case, as long as the personal data is not anonymized, its use needs to be regulated. This paper proposes an automated mechanism for mandatory enforcement of privacy promises given to customers.

A. *Motivation*

The data subject whose personnel data is collected and stored usually has little control over its usage. The notion of privacy when personal data is collected implies some form of trust in the data-collecting entity, but this trust is not necessarily extended to its employees. There is thus a need for privacy protection mechanisms to enforce the privacy promises made by data-collecting organizations to data subjects (e.g. customers).

Furthermore, privacy is very subjective. Different people have different opinions of what is privacy intrusive and what is not, and also on whether an entity is trustworthy or not. In other words, people have different privacy preferences, and should be allowed to express these preferences and have them respected. For an organization having thousands of customers with different privacy preferences, automated solutions for privacy enforcement are necessary.

A system for automated and mandatory enforcement of privacy policies would provide a tool for organizations to enforce the privacy promises given to customers. The implementation of such a system in an organization may contribute to the establishment of trust, and allow individual preferences to be taken into account.

This paper presents Carnival, a framework which, when integrated with applications, provides both access control regulated by privacy policies, and audit functionality to ensure accountability.

Carnival provides functionality for proactive and reactive control to ensure that the purpose of each access corresponds to the purpose stated when the data was collected. It provides a tool for organizations to ensure that their privacy promises are enforced and not breached by individuals associated with the organization.

B. *Outline*

Section II presents some related work in the area of privacy

Manuscript submitted August 2, 2004. The work presented in this paper is fully funded by the Norwegian Research Council through the research project "Personalized Internet-Based Services and Privacy Protection."

R. R. Arnesen is with Norsk Regnesentral (Norwegian Computing Center), P.O.Box 114 Blindern, NO-0314 Oslo, Norway (phone: +47 22852565; fax: +47 22697660; e-mail: Ragni.Ryvold.Arnesen@nr.no).

J. Danielsson, is with Norsk Regnesentral (Norwegian Computing Center), P.O.Box 114 Blindern, NO-0314 Oslo, Norway (e-mail: Jerker.Danielsson@nr.no).

B. Nordlund is with Norsk Regnesentral (Norwegian Computing Center), P.O.Box 114 Blindern, NO-0314 Oslo, Norway (e-mail: Bjorn.Nordlund@nr.no).

access control. How privacy access control differs from "traditional" access control is discussed in section III. Then, in section IV the functionality needed in a framework like Carnival is explored. How this functionality is provided by Carnival is discussed in section V, and section VI explains how Carnival is configured and used. Finally, section VII provides some closing remarks.

II. Related work

In [7] Fischer-Hübner presents a formal task-based privacy model for enforcement of privacy policies and its implementation. The central idea is to control access to personal data through strict control of the tasks users perform. In this setting, a task consists of a set of allowed transformation procedures. Access to personal data is only granted if it is necessary for the task, the user is authorized for the task, and the purpose of the task corresponds to the purpose stated when the information was collected, unless the user has consented to the new purpose.

Karjoth and Schunter present a privacy policy model for enterprises in [10]. They create a privacy control language that includes, among others, user consent, other conditions, and obligations. The policy model allows administration of the system authorizations to be distributed e.g. between a privacy officer and a security officer, while guaranteeing separation of duty.

IBM has developed Declarative Privacy Monitoring (DPM) [5]. DPM is a Java library for adding privacy access control and auditing functionality to J2EE web applications, and hence it can only be applied to applications running in a J2EE context. In contrast, our implementation of Carnival works with plain Java applications. Like Carnival, DPM provides access control based on the action requested, the identity/role of the requesting user, the purpose of the access, the identity and preferences of the data subject associated with the data, and the type of personal data to be accessed. However, DPM does not include any functionality for communicating the current task (i.e. purpose) to the user or functionality for the user to override the current task.

In [2] we present a framework for enforcement of privacy policies. Here we give a description of the functionalities that are necessary to enforce privacy policies and legislation. In the context of this framework Carnival implements the Reference Monitor and it provides a tool for generating the logs that are analyzed by the components of the Monitoring element.

The Hippocratic Database concept is introduced in [1]. It is argued that future database systems should include functionality for protecting the privacy of the data they store. A strawman design for such a database is presented. The design outlines, among others, how queries on the data in the database is regulated according to policy and how information about performed queries are logged.

One main difference between the proposed solutions is at which layer the privacy access control logic is applied. The purpose of an access is easiest determined at the layers closest to the user, whereas the personal data accessed is easiest determined at lower layers. DPM, like Carnival, implements access control in the data layer of the application and determines the purpose of access in higher layers. The Hippocratic Database implements access control in the database layer, and the implementation of Fischer-Hübner's privacy model implements access control in the operating system layer. These two last solutions require applications to propagate the purpose of accesses to the database and the operating system, respectively.

III. Privacy access control

Access control forms a necessary basis for enforcement of privacy, but it is important to realize that *privacy access control* is different from "traditional" access control. This is mainly for two reasons.

First, the *purpose* of data access is important. When personal information is collected, the purpose of the collection must be stated. If a subsequent request for access to the information is made, the purpose of the information access must correspond to the purpose stated when the information was collected. Using the information for other purposes should not be allowed unless the data subject consents, or there is a legal right (or obligation) to do this. These principles can be found in the OECD guidelines [12], and are important in most enacted privacy legislations (e.g. EU Directive [6]). The stated purpose of accesses is up to the discretion of the user and therefore audit is necessary to detect misuse through false purpose statements.

Second, access to personal information could lead to *obligations* that must be addressed. For example, legislation may require that a person should be notified when someone runs a credit check on him, or one may be required to delete or depersonalize information after a given period of time. In many cases, it is not possible to check that the obligations are fulfilled before information access is granted. Hence, proper workflow control and audit of system behavior are crucial to ensure that obligations are indeed fulfilled as required.

Privacy access control is regulated by the rules of a privacy policy. An example of such a rule written in plain English is: "An insurance agent (*role*) may read (*action*) information about my financial situation and living conditions (*data type*) if he uses it to offer me a tailored insurance package (*purpose*), provided that I am notified (*obligation*)". Such rules may be written in a machine-readable policy language, e.g. EPAL [3], for automated evaluation by a rule engine.

Carnival regulates access based on the current purpose of the user. Privacy policies, and the purpose statements they contain, may be rather abstract to be manageable and accessible to humans. Computer applications are generally only aware of what the user wants to do (i.e. the requested operation), not why (i.e. for which purpose). To automatically enforce abstract policies the stated purposes may have to be refined into more concretely defined purposes and these purposes can be associated with the operations of the application.

If individual preferences are to be taken into account, there will be one set of policy rules for each individual in addition to the organization's policy. Thus, the identity of the data subject whose data is requested must be taken into consideration when determining which policy rules to evaluate against the access request. In addition, there may be a need to retrieve and evaluate different types of context information, such as access history, time of day, current location of the data subject, or whether or not a specific relation exists between the user and the data subject. This contributes to the complexity of implementing privacy access control.

IV. Requirements

This section explores some important requirements that apply to privacy access control mechanisms. The subsequent sections describe how Carnival meets these requirements.

To be able to evaluate access requests against a privacy policy, the following information must be retrieved for each access request:

- The identity and/or roles of the user who wants to access the data.
- The action requested on the data.
- The purpose of the access.
- The type of data requested.
- The identity of the data subject.

The identity of the data subject is needed to identify the data subject's individual policy, which formalizes the user's choices, consents and conditions. Note that this identity may be a pseudonym.

Additionally, it may be necessary to provide other information to evaluate deployment specific conditions. For example, the policy of a pharmacy might state that a pharmacist may only access prescriptions if the data subject of the prescription is present (e.g. proven by inserting a smart card into a reader). In this case the location of the data subject is also needed to evaluate access requests.

The access control mechanism must obviously include logic, or connect to logic, for evaluating access requests based on the information above. This evaluation logic should be easily replaceable; it should be possible to plug in evaluation logic implementing different access control models and supporting different policy languages.

Further, plug-ins for executing different types of obligations should be supported. Obligations that should be supported are obfuscation (e.g. making the data less granular) and pseudonymization of data before access is granted.

In addition, the access control mechanism should guarantee that the user and access control mechanism have the same understanding of what the user's purpose is. It is important to ensure that a user cannot make the case that he or she was accessing the data for another purpose than the one registered by the access control mechanism.

Finally, to ensure flexibility, the policy-based access control mechanism should be kept separate from the application code. The access control mechanism should not make any assumptions that the application in any way restricts access to personal data.

V. Carnival

Carnival intervenes in the execution of the application when users access personal data. The central privacy-enforcing element of Carnival is the Privacy Manager that protects data objects containing personal data in the application. In Carnival terminology such objects are called personal data objects.

The Privacy Manger can be seen as a container. Data in this container is privacy protected; data outside this container is not. Objects inside this container are called privacy-managed personal data objects (or just managed objects).

The Privacy Manager intercepts both before and after access to the personal data in managed objects. Before access, logging and access control is performed. After access, logging and obligation execution is performed. The Privacy Manager can be integrated with the application using a number of different techniques. In the current version of Carnival, Dynamic Proxies[1] are used. Another alternative could have been to use Aspects[2].

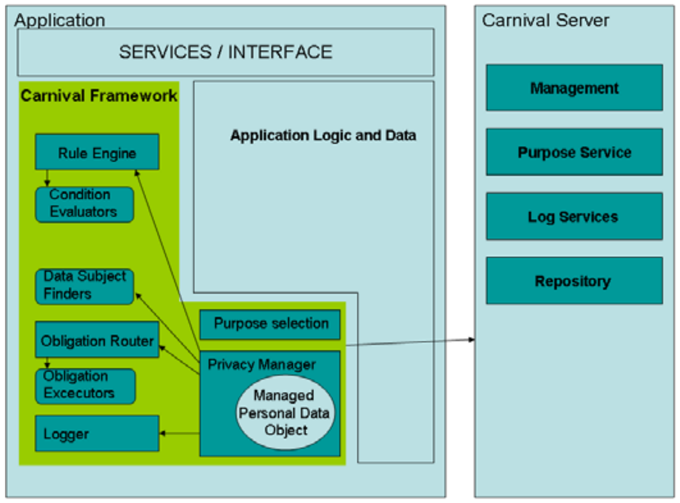

Figure 1 Overview of Carnival

A. Architecture

Carnival consists of the Carnival Framework and the Carnival Server (see Figure 1). The Carnival Framework is integrated into applications and it uses the services of the Carnival Server in the enforcement of the privacy policy of the organization.

The Carnival Framework is made up of the Privacy Manager, a number of services (rectangles in Figure 1), and optional developer- and deployer-supplied plug-ins (rounded corners in Figure 1).

[1] See http://java.sun.com/j2se/1.3/docs/guide/reflection/proxy.html
[2] See http://en.wikipedia.org/wiki/Aspect-oriented_programming

The Privacy Manager intercepts and collects information about access requests to personal data in managed objects. The Privacy Manager uses Data Subject Finder plug-ins to retrieve the identity of the data subjects whose personal data is requested.

The Rule Engine evaluates access requests on behalf of the Privacy Manager. The Rule Engine used by the current version of Carnival evaluates EPAL policies. It is possible to replace this Rule Engine with other implementation possibly supporting other access control models (e.g. Chinese Wall [4]) and/or policy languages. In the work of evaluating an access request the Rule Engine may call one or several Condition Evaluator plug-ins.

Accesses may result in obligations. These obligations are interpreted by the Obligation Router and handed off to the appropriate Obligation Executor plug-in.

Audit logs are created by the Logger service. It receives information about access requests and accesses to managed personal data objects from the Privacy Manager and constructs log records according to the log policy.

Finally, the Purpose Selection service implements logic for determining users' current purposes. A method for determining a user's current purpose is presented in section V.E.

The Carnival Server consists of:

- A Management interface, for managing organization policy (vocabularies, privacy policies, and log policies) and application configuration (links to user directory, purpose rules, and privacy metadata descriptor).
- A Repository, providing the Carnival Framework access to configuration and policy.
- A Purpose service, which stores users' current purposes. A central Purpose service enables the purposes of users to be determined based on their actions in different applications.
- A Log service, which receives and accumulates the logs created by the Logger.

Carnival enforces privacy policies that regulate access based on the action requested, the identity and/or roles of the requesting user, the purpose of the access, the identity and preferences of the data subject associated with the data, and the type of personal data to be accessed. This information is application independent. Hence, unfavorable coupling of policy and applications is avoided. The same organization-wide policy can be applied to all applications without any adaptation to the policy.

The Carnival Server leverages this decoupling between policy and applications by providing central management and integration of policy enforcement in applications. It offers a default implementation of the services that are needed by the Carnival Framework. This reference implementation may be replaced with other solutions implementing the interfaces and services required by the Carnival Framework.

B. *Execution flow*

Carnival regulates access to get and set methods[3] in managed personal data objects. Carnival requires that all access to personal data contained in personal data objects goes through get and set methods.

When data in a managed object is requested the Privacy Manager derives the action requested from the method called and retrieves the purpose and roles of the requesting user from the Carnival Server.

The Privacy Manager also retrieves information about the data subject and the data types of the data to be accessed. This information is collectively termed privacy metadata.

All this information is passed on to the Rule Engine that determines which policy to evaluate the request against. If an individual policy is available, it is used. Otherwise, the default local policy is used.

If the policy contains conditions these are evaluated by Condition Evaluator plug-ins. Condition Evaluators receive information about the access request (user, data subject, etc) from the Rule Engine. If the Condition Evaluator needs other information for evaluating the condition the Condition Evaluator retrieves this information from the application or some external information source.

The Rule Engine may decide that the user is denied access or, alternatively, that the access is granted. If access is denied, an exception is thrown, which the application should take appropriate actions to handle, e.g. roll back transactions and/or provide a message to the user.

The Rule Engine may associate the grant to access with one or several obligations, as determined by the applicable policy. If so, the obligations are passed on to the Obligation Router by the Privacy Manager. The Obligation Router routes the individual obligations to the correct Obligation Executor instances.

There are two types of Obligation Executors: synchronous and asynchronous. Synchronous Obligation Executors block until a result is returned. The Rule Engine may, for example, demand that, before access is granted, the level of detail in the result should be reduced according to a specification provided by the policy. For example, the age of the data subject may be replaced by an interval.

Asynchronous Obligation Executors do their job in the background. An example of such an Obligation Executor is one that is capable of sending notifications to data subjects through email.

The rest of this paper focuses on the access control functionality of Carnival. Under the hood the logging functionality is implemented much the same way as the access control functionality. The main difference is that in the case of logging, information collected is used to create a log record that is sent to the Log Service, whereas for access control the information is sent to the Rule Engine for evaluation. It is important that the log is subjected to manual and possibly

[3] A get method (e.g. getId) of an object retrieves the value of an instance variable (id) of the object. A set method modifies the value of an instance variable of an object.

automated audit to detect privacy violations.

C. *Privacy metadata*

The type of data to be accessed and whom the data is about are natural to extract from the objects in the application that represent the data subjects and that consequently contain information pertaining to data subjects. For example, in an Electronic Patient Journal (EPR) application it is natural to extract this metadata from the objects in the application that represent patients.

Consequently, Carnival introduces the concept of personal data classes and objects. Personal data classes are classes that define instance variables that hold personal data and where one or more of these instance variables can be used to identify the data subject of the contained personal data. Personal data objects are instances of personal data classes.

Metadata must be provided for all personal data classes in the application. The metadata serves two purposes, it defines: (i) which types of personal data that the get and set methods of the personal data classes return and modify; (ii) how the data subject of a personal data object can be determined during runtime.

```
<?xml version="1.0"?>
<application name="EPR" >

    <vocabulary name="epr-voc.xml" >

   <personaldataclasses>

      <class name="epr.model.Pasient" managed="y">
            <datasubject>
                <finderclass classname="epr.subfinder.Journal" />
             </datasubject>
             <property name="id" >
                <type name="PERSON_ID" />
             </property>
             <property name="firstName">
                <type name="PERSON_NAME"/>
             </property>
           . . .
      </class>
           . . .
   </persondataclasses>
</application>
```

Figure 2: Example Metadata mapping file

The metadata maps the application data to a vocabulary so that the policy written in this vocabulary can be interpreted and enforced in the context of the application.

The vocabulary defines the entities (i.e. words) of the language used to express privacy policies. It defines valid data types (e.g. first_name), purposes (e.g. diagnosis), and actions (e.g. read, write). The application developers are free to choose a suitable vocabulary, preferably a standardized vocabulary for the application domain, if available.

Privacy metadata is supplied through metadata descriptor files, one file for each application. These files can be edited directly or through the Management interface of the Carnival Server.

Figure 2 shows an excerpt of the metadata descriptor file for the *EPR* application. It identifies the vocabulary used and the personal data classes of the application. The *Patient* class contains (at least) two instance variables holding personal data. The *id* instance variable is of type *PERSON_ID* and the instance variable *firstName* is of type *PERSON_NAME*. In addition there is a reference to the Data Subject Finder plug-in, *epr.subfinder.Journal*, which is used to identify the data subject of an instance of the class.

The metadata descriptor is created during application development and it may be edited during application deployment. Among others, during deployment it is determined which personal data classes that should be managed. How applications using Carnival are deployed and configured is described further in section VI.

Privacy metadata can also be provided through code annotations or through annotations of UML-diagrams constructed during the design phase. From these annotations the metadata descriptor of the application can be automatically generated. Figure 3 shows Java 1.5 annotations corresponding to the metadata descriptor file in Figure 2.

```
@no.nr.privacy.annotations.DataSubjectHelper
 ("epr.subfinder.Journal")
public class Patient{

      @no.nr.privacy.annotations.PersonalDataType("PERSON_ID")
      private int id;

    @no.nr.privacy.annotations.PersonalDataType("PERSON_NAME")
      private String firstName;
                 .
      public int getId() {
         return id;
      }
      public String getFirstName() {
         return firstName;
      }

    }
```

Figure 3: Example annotated class

D. *Extraction of privacy metadata during runtime*

When the Privacy Manager evaluates an access request to personal data contained in a managed object the metadata of the requested data is retrieved. The types of the data requested is a static property, whereas the identity of the data subject is a dynamic property that can only be determined at runtime.

When a managed object is accessed the data types and its Data Subject Finder plug-in are looked up in the metadata descriptor file. Finally, the personal data object is passed to the Data Subject Finder plug-in that returns a string that identifies the data subject.

E. *Purpose selection*

The Purpose Selection service implements Carnival's purpose selection logic. The Purpose Selection service's behavior is defined by purpose rules. In the current implementation a user's current purpose is determined as a function of the user's roles and the method invoked by the user. The Purpose Selection service collects this information before method invocations.

The application should provide methods that are called when the user moves from one purpose to another. One way of accomplishing this is to design the application so that each task in the application is naturally delimited from the other tasks, for example through providing different GUI views for each task.

The user and application must of course have the same understanding of what the current purpose is. One way to achieve this is to have the application clearly display the current purpose and require that the user actively change this purpose if he/she disagrees. Carnival requires that applications provide Carnival with some method of communicating directly with users. More precisely, Carnival requires that applications provide callbacks, which Carnival uses to present the user's current purposes, and functionality for actively changing the current purpose.

VI. USAGE

The usage of Carnival can be divided into three phases: development, deployment, and operation.

A. *Development*

When developing an application using Carnival some design guidelines should be followed.

The application must take into consideration the fact that access to a method can be denied leading to an exception being thrown. Likewise, when an access has lead to obligations this is communicated to the application through an exception. This exception contains information about the executed obligation and the result of the access, which may have been affected by the obligation. For example, an approximation may be returned instead of the exact value. The information contained in exceptions allows the application to communicate to the user why access was denied and/or which obligations that have been executed.

Additionally, as stated before, the application should be designed in such a way that it is easy to capture the current purpose of users. Carnival also requires that Data Subject Finder plug-ins and GUI callbacks for purpose management are developed. Developers may also supply Condition Evaluators and Obligation Executors relevant for the application domain.

Furthermore, the privacy metadata descriptor file should be written, listing all personal data classes of the application, as described in section V.C.

B. *Deployment*

During application deployment, a vocabulary must be constructed or selected if one does not exist (see section V.C). The vocabulary used by the application may be adopted, a new may be constructed, or a standard vocabulary may be adopted. If the vocabulary bundled with the application is not used, the application's metadata descriptor file needs to be updated so that it is compliant with the new vocabulary.

Additionally, Carnival must be provided with a local privacy policy conforming to the chosen vocabulary, if not already available. For all obligation types included in the local policy, corresponding Obligation Executor plug-ins should be provided, and for each type of condition in the policy, a corresponding Condition Evaluator should be provided.

Finally, two application specific steps should be followed. Firstly, it should be decided which personal data objects that should be managed from the ones listed in the privacy metadata descriptor. Secondly, purpose rules should be provided (see section V.E).

C. *Operation*

When a relationship is established with a data subject (e.g. customer) his or her privacy preferences may be taken into account by creating an individual privacy policy for the data subject. This policy is added to the Carnival's repository of policies to be enforced.

VII. CONCLUDING REMARKS

This paper has presented Carnival, a framework that provides privacy protection functionality to applications. Carnival enables the implementation of privacy access control, which is different from the other types of access control, as discussed in section III. In addition, it provides functionality for producing audit trails to enable detection of privacy violations. Finally, it defines a number of services and plug-ins to support the enforcement of privacy policies: The Purpose and Purpose Selection services which provide information needed to evaluate policy rules, the Rule Engine and the Condition Evaluator which evaluate access requests to personal data, the Obligation Router and Executors which enforce obligations resulting from data access, and the Logger and Log services which handle audit trails.

However, note that Carnival does not include all functionality needed to enforce privacy policies. Organizations also need, among others, to provide channels for data subjects to access their personal data and data about its usage (Individual Participation Principle, see [12]), and measures to continually uphold the accuracy and completeness of the personal data stored (Data Quality Principle, see [12]).

Carnival fulfils the requirements listed in section IV, except support for pseudonyms, which has not been implemented yet. That is, it enables the retrieval of all information needed to evaluate privacy policy rules and determine whether or not requested access should be granted. Further, the access control logic is replaceable, and hence supports the implementation of different access control models and the use of different policy languages. In addition, services are defined to support different types of obligations.

The determination of a user's current purpose is handled through the inclusion of the Purpose Selection service. However, the design of this service needs to be further explored, as the determination of the current purpose is an intricate problem. We are not convinced that we have seen the best solutions to this problem yet. One possibility we will investigate further is the use of a formal workflow-control system based on the use of Petri nets (see e.g. [9]). An advantage of this type of solution is that purposes can easily be defined across different applications.

Going forward, we plan to add support in Carnival for creation and management of pseudo domains, as proposed in [8]. Carnival will thus include functionality for generating pseudonyms, and regulating linkage between pseudonyms and the disclosure of the identities behind pseudonyms. This functionality is motivated by the fact that some functions in an organization may not need to have knowledge of information that directly identifies data subjects, typically the data subjects' name or identity number. Their work can equally well be carried out when data subjects are identified by pseudonyms. Additionally, different functions can be provided with different pseudonyms for the same data subject, preventing unauthorized linking and matching of information.

We also plan to further evaluate the usefulness and performance of Carnival. Questions related to how intuitively it is to integrate with applications, how well it scales, and how it affects performance, will be further examined.

VIII. REFERENCES

[1] R. Agrawal, J. Kiernana, S. Ramakrishnan, and Y. Xu, Hippocratic Databases, IBM Almaden Research Center. Available at: http://www.almaden.ibm.com/software/dm/Hippocratic_Databases/hippocratic.pdf

[2] R. R. Arnesen and J. Danielsson: "A Framework for Enforcement of Privacy Policies," in Proceedings of the Nordic Security Workshop NORDSEC 2003, October 2003. Available at: http://publications.nr.no/A_Framework_for_Enforcement_of_Privacy_Policies.pdf

[3] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter (ed.), Enterprise Privacy Authorisation Language (EPAL 1.1), IBM, 2003. Available via http://www.zurich.ibm.com/security/enterprise-privacy/epal/

[4] D.F.C. Brewer and M.J. Nash, "The Chinese Wall Security Policy," IEEE Symposium on Security and Privacy, pp. 215-228, 1989

[5] Declarative Privacy Monitoring, IBM alphaWorks, http://www.alphaworks.ibm.com/tech/dpm

[6] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. Official Journal L 281, 23/11/1995, pp. 31-50. Available from http://europa.eu.int/eur-lex/en/index.html

[7] S. Fischer-Hübner and A. Ott., "From a Formal Privacy Policy Model to its Implementation," National Information Systems Security Conference (NISSC 98), 1998. Available at http://www.rsbac.org/niss98.htm

[8] R. Hes and J. Borking, (eds.), Privacy-enhancing technologies: The path to anonymity, Revised edition. ISBN: 90-74087-12-4. Registratiekamer, The Hague, August 2000

[9] K. Jensen, "Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use, Volume 1, Basic Concepts." Monographs in Theoretical Computer Science, Springer-Verlag, 1997

[10] G. Karjoth and M. Schunter, A Privacy Policy Model for Enterprises, 15th IEEE Computer Security Foundations Workshop, June 2002

[11] M. Hansen and A. Pfitzmann, Anonymity, Unobservability and Pseudonymity – A Proposal for Terminology, v0.21, Available at http://dud.inf.tu-dresden.de/Literatur_V1.shtml

[12] OECD, Guidelines on the Protection of Privacy and Transborder Flows of Personal Data, Available at http://www1.oecd.org/publications/e-book/9302011E.PDF