Enhancing User Privacy Through Data Handling Policies

C.A. Ardagna, S. De Capitani di Vimercati, and P. Samarati

Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano – 26013 Crema - Italy
{ardagna, damiani, decapita, samarati}@dti.unimi.it

Abstract. The protection of privacy is an increasing concern in today's global infrastructure. One of the most important privacy protection principles states that personal information collected for one purpose may not be used for any other purpose without the specific *informed consent* of the person it concerns. Although users provide personal information for use in one specific context, they often have no idea on how such a personal information may be used subsequently.

In this paper, we introduce a new type of privacy policy, called *data handling policy*, which defines how the personal information release will be (or should be) dealt with at the receiving party. A data handling policy allows users to define simple and appropriate levels of control over who sees what information about them and under which circumstances.

1 Introduction

Privacy is repeatedly identified as one of the main concern that prevents users from using the Internet for transactions. Information technology gives organizations the power to gather and disclose vast amounts of personal information and therefore those who collect and disseminate data should be responsible for maintaining privacy. A number of useful *privacy enhancement technologies* (PETs) have been developed for dealing with the privacy issue such as a variety of anonymizing and de-identifying mechanisms [10]. One of the most important privacy protection principles (see `http://ohsr.od.nih.gov/guidelines/belmont.html`) states that personal information collected for one purpose may not be used for any other purpose without the specific *informed consent* of the person it concerns. However, although the informed consent is intended to prevent inappropriate use of the data, it represents an important problem involving PETs because users provide personal information for use in one specific context, but they often have no idea on how a such personal information may be used subsequently. In other words, users do not always realize that the information they disclose for one purpose (e.g., name, date of birth, and address within an on-line transaction) may also have secondary uses (e.g., access to existing data for purposes of grouping together users on the basis of common characteristics such as age or geographic

E. Damiani and P. Liu (Eds.): Data and Applications Security 2006, LNCS 4127, pp. 224–236, 2006.

location). Therefore, even if users consent to the initial collection of their personal information, they must also be given a mechanism to specify whether or not to consent to the future use of that information in secondary applications.

In this paper, we focus on a new type of privacy policy, called *data handling policy*, that regulates the secondary use of a user's personal data. In particular, a data handling policy regulates how Personal Identifiable Information (PII) will be used (e.g., information collected through a service will be combined with information collected from other services and used in aggregation for market research purposes), how long PII will be retained (e.g., information will be retained as long as necessary to perform the service), and so on. Users can therefore use these policies to define how their information will be used and processed by the counterpart.

2 Related Work

Previous work on privacy protection has focused on a wide variety of issues [4,9,14,18,21]. The work most directly related to ours is in the area of access control and privacy-aware languages and models [2,7,13,8,12,20]. In [8] the authors propose a policy language for regulating service access and information disclosure in an open, distributed network system. Access regulations are specified as logical rules, where some predicates are explicitly identified. Besides certificates, the proposal also allows to reason about declarations (i.e., unsigned statements) and user-profiles that the server can maintain and exploit for taking the access decision. PeerTrust [12] defines a logic syntax used to automate trust establishment. Trust is established gradually by disclosing certificates and requests for certificates. Each party can define access control policies to protect their sensitive resources. PROTUNE (PROvisional TrUst NEgotiation) [7] is a policy specification language that provides a powerful declarative metalanguage for driving critical negotiation decisions such as the specification of what certificates are needed to gain an access, where certificates can be retrieved, and so on. In [2], the authors propose an XML-based privacy preference expression language, called *PReference Expression for Privacy* (PREP), for storing the user's privacy preferences with Liberty Alliance. PREP allows users to specify, for each attribute, a *privacy label* that is characterized by a purpose, type of access, recipient, data retention, remedies, and disputes.

P3P (Platform for Privacy Preferences) [20] is an XML-based language that addresses the need of a user to assess whether the privacy practices adopted by a server provider comply with her privacy preferences. Users specify their privacy preferences in term of a policy language, called APPEL [19], and enforce privacy protection through a user agent. The user agent compares the users' privacy policy with the service provider P3P policy and checks whether the P3P policy conforms to the user privacy preferences. Although P3P is a good starting point, it has some drawbacks as the lack of a technical mechanism to verify that Web sites respect and enforce users policies, and a process to negotiate the privacy practices between the interacting parties. Also, P3P presents some

limitations on the user side [1]: users can only accept or deny the privacy practices defined by a service provider. We believe that a better way to enforce the privacy practices is to offer users a richer, more active role in establishing how their personal data should be used. Two relevant XML-based languages designed to enforce privacy policies are XACML (eXtensible Access Control Markup Language) with a privacy policy profile [11,15] and EPAL (Enterprise Privacy Authorization Language) [5]. They allow the definition of powerful and expressive access control languages but do not regulate the use of personal information in secondary applications.

3 Data Handling Policy Specification

We consider a scenario that involves three main entities: *users* are human entity that present requests to the service provider; a *service provider* provides services and collects personal information from users; and *third parties* are external organizations to which the service provider can disclose personal information. We assume that the service provider collects personal data that are necessary to provide access to services. In particular, when a user decides to use a service, she needs to complete a registration process. Information collected from a given user is then stored into *profiles* associated with the user. Registered users are characterized by a unique *user identifier* (user id). Users may also choose not to become registered users. In this case, the service provider can generate a *persistent user identifier* (pseudonym) that is associated with the user who requires the service. The pseudonym is automatically sent by the Web browser to the service provider whenever the user submits a request to the service provider.[1] In this case, personal information is stored under pseudonyms and not users' real names. Users can require access to data about themselves. Other access to personal data by the employees of the service provider (internal users) and third parties who are granted access by the service provider (external users) should also be supported. For simplicity, we assume that the server collecting the users data and the third parties accessing them are trusted entities. Personal data collected by the service provider should be managed in accordance with the informed consent principle stating that personal information will not be made available for secondary uses without notice to the subjects of the information. However, there are situations where the strictly application of this principle can be impracticable to enforce (e.g., in the context of large studies on population health). A possible approach to solve this problem consists in giving the users the possibility to specify a policy, called *data handling policy*, which defines how their data can be subsequently used by the service provider and/or third parties. The data handling policy follows the data when they are manipulated by different applications and transferred between different systems. A data handling policy should be simple and expressive enough to support the following privacy requirements.

[1] This features can be implemented using different strategies (e.g., cookies). However, this implementation issue is outside the scope of our paper.

- *Individual control.* Users should be able to specify who can see what information about them and when.
- *Consent.* Users should be able to give their explicit consent on how to use their personal data.
- *Correction.* Users should be able to access their personal information to modify it when needed.
- *Security.* Adequate security mechanisms have to be applied, according to the sensitivity of the data collected.

Data handling policies can be pre-defined by service providers (and possibly by users) or can be defined at access time. These different strategies require different levels of negotiation between a user and a service provider. In particular, we identify the following three strategies: *server-side* strategy, where a service provider defines its data handling policies and a user can accept or reject these policies according to her privacy preferences; *customized*, where a user requires a service and a predefined policy template is provided by the service provider as a starting point for creating data handling policies; *user- and server-side*, where both a user and a service provider define their data handling policies and a negotiation process between them starts. This negotiation process can be initiated either by the service provider or the user. The negotiation process ends when the involved parties have reached an agreement. The user then provides her personal data attached to the data handling policy on which the user and the service provider agree.

Another aspect that has to be investigated is how data handling policies can be integrated with traditional *access control policies* [17]. Intuitively, if an access request satisfies at least one access control policy, the service provider has to verity whether there exists at least one data handling policy attached to the requested data. In particular, there may exist one or more data handling policies and each of them can impose different restrictions on how such data can be used in secondary applications. Since, as we will see in Section 4.1, a data handling policy establishes which party (*subject*, in the access control terminology) can execute which *actions* on which *resources* and under which circumstances (*conditions*), it is easy to see that access control and data handling policies are similar in syntax but they are conceptually different. Data handling policies allow the users to define restrictions on their PII management when data are received at server-side (e.g., retention, notification, and so on), while access control policies protect access to data. At evaluation-time, data handling policies are evaluated together with the access control policies, but at disclosure time they are attached to the released data, building a chain of control coming from the data owner. The similarity with access control policies introduces two different ways for defining a data handling policy: it can be an extension of traditional access control policies or it can be defined as a *stand-alone* policy. In the first case, the authorization rules should be extended by adding a *DHP component*. This approach has the main disadvantage that whereas including a DHP component within an access control rule simplifies the policy specification at first sight, it also makes the policy less clear. Stand-alone definition means that data handling policies are

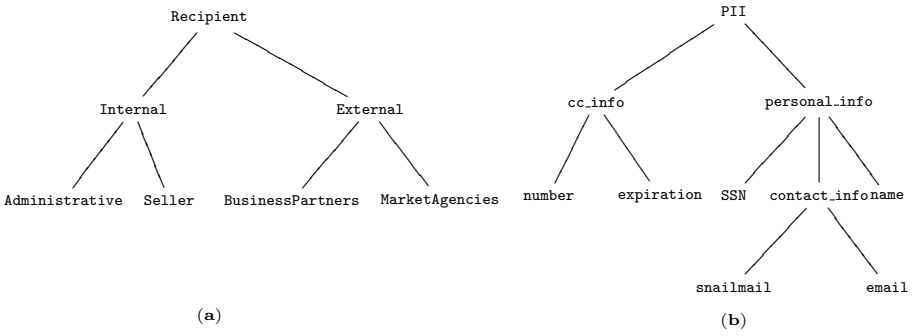

Fig. 1. An example of recipient hierarchy (a) and data type hierarchy (b)

defined as independent rules. Therefore, a data handling policy should represent the users' privacy preferences and should then include different components that allow to define how the third parties can use personal data. Personal data are then *tagged* with such data handling policies. Although the stand-alone option can introduce some redundancy in policy definition, choosing the stand-alone representation provides a good separation between policies that are used with different purposes. This clear separation makes data handling policies more intuitive and user-friendly, and implicitly suggests the differences with access control policies.

4 DHP Language

We now introduce a language for the specification of data handling policies on the data. We start by describing the basic constructs of the language and then we illustrate the syntax of the language together with some examples.

4.1 Elements of a Data Handling Policy

To illustrate what kinds of privacy requirements our solution supports, in the following we will consider an e-commerce scenario where company `ACME` provides a set of services such as *rent-a-car*, *book-a-flight*, and *flight+hotel*.

Recipients. It is a third party to which PII can be disclosed. Since we work in an open environment, the third parties may be unknown a-priori and therefore the user should also have the ability to define to which entities her data may be disclosed without knowing their identity. Our approach supports then the definition of the recipient according to one of the following three options: *identity-based*, where the third parties may be identified by their unique identities; *category-based*, where the third parties are grouped into different categories, which represent recipients of different domains; and *attribute-based*, where an *attribute expression* allows the definition of a recipient in term of the properties that it has to satisfy, instead of its identity.

Categories can be hierarchically organized and within the hierarchy, a category inherits from its ancestors all their permissions. For instance, Figure 1(a) illustrates a recipient hierarchy where the third parties are partitioned into two main domains: `Internal` and `External`. Also, the attributes that characterize a recipient can be *certified* by given authorities or can be simply *declared* by the recipient itself. At an abstract level, an attribute certificate is characterized by the following elements: the certificate's *name*, the *issuer*'s public key, the *subject*'s public key, a *validity period*, a list (possibly empty) of pairs ⟨*attribute_name*, *attribute_value*⟩ representing the subject'attributes, and a *signature*. For simplicity, we assume that the service provider maintains a binding between the public key of the authority trusted for asserting a specific set of attributes and an authority name. For instance, *InternationalMarketBoard* is the name of the authority whose public key is `00:b4:31...:7e:41:8f`. Note that in case of certified attributes, the user can then define the authorities trusted for asserting those attributes. For instance, a user can decide to disclosure her personal information only to a company of the `MarketAgencies` category specialized for distribution of computers and such that its attribute specialization has been certified by the *InternationMarketBoard* authority.

Actions. The term *action* is used to denote privacy-relevant operations (e.g., `read`, `disclose`, and `modify`) that recipients can require on personal data.

Privacy profiles. Users interacting with a service provider are required to provide a significant amount of PII, which is stored within a set of *privacy profiles*. A privacy profile can be seen as a container of pairs of the form ⟨*attribute_name*,*attribute_value*⟩, where *attribute_name* is the name of the attribute provided by the user and *attribute_value* is the value. Each privacy profile, uniquely identified by the user id or pseudonyms together with a *profile number*, is characterized by a specific set of privacy preferences expressed via data handling policies. For instance, suppose that `Alice` is a registered user of company `ACME` and that she requires service *book-a-flight*. To book a flight, `Alice` must provide her `name`, `credit card number` and `expiration date`, `telephone number`, `e-mail address`, and `frequent traveler number` (if any). Since `Alice` requires different levels of privacy according to her perception of the information sensitivity, her PII is partitioned into two privacy profiles: profile `p1` stores the `credit card number` and `expiration date`, and the `telephone number`; profile `p2` stores the `name`, `e-mail address`, and the `frequent traveler number`. Intuitively, the first profile includes information with a high level of sensitivity and the second profile stores information with a lower level of sensitivity. A data handling policy can be defined on a whole privacy profile and/or can be associated with a specific attribute in a privacy profile. Data types can be introduced as abstractions on single PII and therefore privacy preferences may be expressed in terms of data types. Data types can be organized into a hierarchy (we assume that users know the data type hierarchy defined by a server). Figure 1(b) illustrates an example of data type hierarchy, where PII has been partitioned into two data types: `cc_info` is an abstraction

for the credit card information and `personal_info` is an abstraction for the personal information. According to the *correction* principle mentioned in Section 3, users can view, update, or delete their personal information (i.e., the information stored in all privacy profiles associated with them).

Restrictions. A privacy statement specifies restrictions that have to be satisfied before access to personal data is granted and such that, if at least one condition is not satisfied, the access should not be granted. One of the most important restrictions is the *purpose* for which the information will be used. Abstractions can be defined within the domain of purposes, which allow grouping together purposes with common characteristics and referring to the whole group with a name (e.g., `pure research` and `applied research` can be seen as a specialization of `research`). We distinguish between two kinds of conditions: *provisions* and *obligations*. Provisions represents actions that have to be performed before a decision can be rendered [6]. For instance, a data handling policy can state that a business partner can read the email address of the users provided that it has *paid a fee*. Obligations represents actions that have to be performed after an access has been granted [6]. For instance, a data handling policy can state that users will be notified whenever their personal information is disclosed. In addition, *generic* conditions evaluate membership of requesters and personal data into classes or properties in their profiles or can represents conditions that can be brought to satisfaction at run-time processing of the request.

Conditions on properties of a party are specified via a set of predicates based on the attribute certificates above-mentioned. Let $\mathcal{C}$ and $\mathcal{P}$ be a set of certificates' names and predicates, respectively. We first need to introduce the concept of *certificate expression* as follows.

Definition 1 (Certificate expression). Given a certificate name $c \in \mathcal{C}$, a *certificate expression* over c is a boolean formula of terms of the form *c.attr_name math-op value*, where *c.attr_name* denotes attribute *attr_name* within certificate c, *math-op* is a standard binary built-in mathematic operator (i.e., $=$, $\neq$, $>$, $\geq$, $<$, $\leq$), and *value* is a constant or an attribute.

A binary predicate `certificate(`*ce*`,`*A*`)` $\in \mathcal{P}$, where *ce* is a certificate expression and A is the name or the public key of a trusted authority is evaluated to true if and only if there exists a certificate c issued by authority A and such that certificate expression *ce* is evaluated to true. Provisions and obligations are represented by two disjoint sets of non predefined predicates $\mathcal{PR} \subseteq \mathcal{P}$ and $\mathcal{O} \subseteq \mathcal{P}$, respectively. Examples of provision predicates are `fill_in_form()` and `log_access()`. Examples of obligation predicates are `notify()` and `delete_after(`*num_days*`)`.

4.2 Syntax

Syntactically, a data handling policy has the form:

⟨*recipients*⟩ CAN ⟨*actions*⟩ FOR ⟨*purpose*⟩ ON ⟨*PII*⟩ [IF ⟨*gen_conditions*⟩] [PROVIDED ⟨*prov*⟩] [FOLLOW ⟨*obl*⟩],

where *recipients* identifies the parties to which the policy refers; *actions* is the action (or class of actions) to which the policy refers; *PII* identifies the personal data to which the policy refers; *gen_conditions* is an optional boolean expression of conditions that every request to which the policy applies must satisfy; *prov* is an optional boolean expression of provisions; and *obl* is an optional boolean expression of obligations that the server must follow when manages the PII.

A data handling policy specifies that *recipients* can execute *actions* on *PII* for *purpose* provided that *prov* is satisfied, *gen_conditions* are satisfied, and with obligations *obl*. The *actions* field in the policy is simply the identifier of an action or group thereof. Data handling policies referred to groups of actions are considered applicable to all actions in the set. We now look at the different components in the rule.

Recipients. The field *recipients* can be an identifier, a recipient category, or a *recipient expression* of terms that evaluate conditions on the requester. A recipient expression is a boolean formula of terms of the form:

- `certificate(`*ce*`,`*A*`)`, where *ce* is a certificate expression over certificate *c* and *A* is the authority that must have issued certificate *c*. The requester and the subject of certificate *c* have to be the same.
- *attr_name math_op attr_value*, where *attr_name* is an attribute, *math_op* is a standard binary built-in mathematic operator, and *attr_value* is a constant or an attribute. Intuitively, this formula is evaluated to true whenever the requester presents property *attr_name* which satisfies the specified condition.

Since it may be necessary to refer to the user of the request being evaluated, we introduce the keyword **requestor**, which is intended to be substituted with the actual parameters of the request in the evaluation at access control time. For instance, recipient **requestor**`.country` = 'EU' indicates that property `country` provided by the party whose request is being processed has to be equal to 'EU'.

PII. The field *PII* can be the name of an attribute, the name of a data type, the identifier of a privacy profile, or a specific attribute stored within a privacy profile, which is specified by means of the usual dot notation (e.g., `Alice.p2.email`).

Prov, Gen_conditions, Obl. These fields contain conditions that are syntactically similar and correspond to a boolean formula of terms of the form:

- `predicate_name(`*arguments*`)`, where *arguments* is a list, possibly empty, of arguments on which predicate `predicate_name` is evaluated.

From the evaluation point of view, however, *gen_conditions*, *prov*, and *obl* are different: provisions are preconditions that need to be evaluated as prerequisites before a decision can be taken; generic conditions specify conditions of different type (e.g., trusted-based, location-based [3], and so on); obligations are additional steps that must be taken in account after the policy evaluation.

As an example of data handling policies, consider the following rules that regulate the secondary use of personal information stored by the `ACME` organization.

Rule 1. The business partners of ACME can read for market purpose the name of the ACME's users provided that they have paid a fee.
BusinessPartners CAN read FOR market ON name PROVIDED pay_a_fee()

Rule 2. The credit card information of Alice can be read by the business partners of ACME for service release purpose and must be deleted at the end of the service.
BusinessPartners CAN read FOR service_release ON Alice.p1 FOLLOW delete_after_service()

Rule 3. The market agencies specialized for distribution of computers and whose specialization has been certified by the *International Market Board* (IMB) authority can read the snailmail information for market purpose.
MarketAgencies AND certificate(speciality.category = 'computer',IMB) CAN read FOR market ON snailmail

Rule 4. The seller of ACME can read the personal information of their clients for statistical purpose during the working hours (i.e., from 8:30 am to 6:00 pm) provided that the access is logged.
Seller CAN read FOR statistical ON personal_info IF time(8:30,6:00) PROVIDED log_access()

Rule 5. The e-mail address of Alice can be released for market purpose to European business partners of organization ACME with the obligation of notify Alice.
BusinessPartners AND **requestor**.country = 'EU' CAN read FOR market ON Alice.p2.e-mail FOLLOW notify()

Rule 6. The administrative staff of ACME can read the contact information of their clients for market purpose only if they are in the building of ACME.
Administrative CAN read FOR market ON contact_info IF inarea(**requestor**, ACMEBuilding)

Rule 1, Rule 3, Rule 4, and Rule 6 are associated with the privacy profiles of the ACME's users that store property name, properties of data type snailmail, properties of data type personal_info, and properties of data type contact_info, respectively. Rule 2, and Rule 3 are associated with and protect the privacy profiles (p1, p2) of user Alice.

5 The Privacy Architecture

We are currently developing a privacy-aware architecture (see Figure 2) in the framework of the European PRIME project [16]. The architecture is composed of three main components: a *Privacy Control Module*, a *Policy Repository*, and a *Context Manager*.

The Policy Repository contains the policies, both access control and data handling policies, used to protect the data/services. It provides functionalities for administering policies such as search, modify, insert, and delete.

The Privacy Control Module operates on top of the context manager and contains two sub-modules: a *Policy Decision Point* (PDP) and a *Policy Enforcement Point* (PEP). PDP is responsible for taking an access decision for all

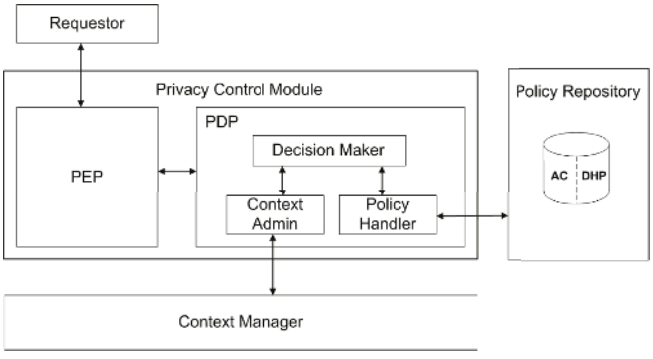

Fig. 2. Privacy-aware access control architecture

access requests directed to data/services. It retrieves and evaluates all policies (both access control and data handling) applicable to a request and includes: a *Decision Maker* that produces the final response possibly combining different access decisions coming from different evaluation of different policy types; a *Policy Handler* that is in charge of managing all communications with the *Policy Repository* to retrieve all policies applicable to an access request; and a *Context Administrator* that manages the access and the communication with the *Context Manager* component, which contains the information associated with the requestor during a session. PEP is responsible for the enforcing of access control decisions by intercepting accesses to resources and granting them only if they are part of an operation for which a positive decision has been taken.

The Context Manager is the component that keeps track of all contextual information and combines information from various context sources and deducts new contextual information from this aggregation. The main task of the Context Manager is to provide contextual information from various sources in a standardized way.

5.1 Access Request Enforcement

We now discuss how our system evaluates an access request and we start by characterizing the requests to which the system will have to respond. Each request is characterized by three elements: the *requestor* that makes the request; the *action* that is being requested; and the *target* on which the requestor wishes to perform the action. We assume that the *requestor* element contains the identity of the requestor (if any) plus additional contextual information such as the purpose for which the access is requested and certificates that can be used to verify whether the requestor has some properties. Such an information provided by the requestor is then managed by the Context Manager. We assume that the target of an access request can be a service or some personal data associated with a specific user. The access request is first received by the PEP module that sends it to the PDP module. The evaluation process is now composed of two main steps.

Step 1. The access request is evaluated against the applicable access control policies. Note that, if no policy is selected the access is denied meaning that the default access decision is "no". The discussion on how this step is performed is outside the scope of this paper and we simply assume that at the end of this first step the system has reached a "yes" or "no" access decision.[2] In case of a negative (no) access decision, the access request is denied and the process terminates. In case of a positive (yes) access decision, the system has to verify whether there exists some restrictions on the secondary use of the requested target.

Step 2. The PDP module asks the Policy Repository to retrieve all the applicable data handling policies. This selection is performed by using the requestor, action, and target specified in the access request. For each applicable data handling policy, the system evaluates the conditions specified in the *gen_conditions* field, when possible. Indeed, field *gen_conditions* can contain conditions that have to be brought to satisfaction at run-time, while processing the request. For each of them, we require the existence of an interface function that performs the control and possibly triggers the necessary actions. The corresponding procedure returns either a true or a false value depending on whether or not the implemented condition was or has been brought to satisfaction. Then, the *gen_conditions* are simplified using the usual boolean laws for true and false and the corresponding policy is taken into consideration if and only if the *gen_conditions* would be simplify to true. At this point of the evaluation process, there may exists different data handling policies with different sets of provisions and obligations. The system now should select the most convenient combination of provisions and obligations for the requestor.[3] As an example, suppose that `Alice` requests service *rent-a-car* and that she has to provide her `name`, `credit card number`, and `expiration date`. Suppose also that `ACME` collaborates with company `BestCar.com` which is a business partner. To make a reservation and rent a car, company `BestCar.com` requires access to the credit card information of `Alice`. By considering the data handling policies in Section 4.2, it is easy to see that Rule 2 applies to the access request submitted by `BestCar.com`. According to this policy, `BestCar.com` can use the credit card information of `Alice` but such an information has to be deleted at the end of the transaction.

6 Conclusions

Privacy is one of the most important issue for electronic commerce. In this paper, we introduce the definition of data handling policies, that is, policies regulating the use of personal information in secondary applications. This paper is only the first step towards the definition of such a language and leaves space for further work. Future work to be carried out includes investigation of the negotiation

[2] Note that a yes/no access decision can be the result of a multi-step process where the requestor and the system interact thus introducing possible forms of negotiation between them.

[3] The development of an efficient technique used to make such a selection will be part of future work.

process between a user and a service provider needed to reach an agreement on the data handling policies; techniques for linking data handling policies with the corresponding personal information; and the implementation of a proof-of-concept prototype, which is under development, to assess the real applicability of the proposed model.

Acknowledgements

This work was supported in part by the European Union within the PRIME Project in the FP6/IST Programme under contract IST-2002-507591 and by the Italian MIUR within the KIWI and MAPS projects.

References

1. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. An xpath based preference language for P3P. In *Proc. of the 12th International World Wide Web Conference*, Budapest, Hungary, May 2003.
2. Gail-J. Ahn and J. Lam. Managing privacy preferences in federated identity management. In *Proc. of the ACM Workshop on Digital Identity Management (In conjunction with 12th ACM Conference on Computer and Communications Security)*, Fairfax, VA, USA, November 2005.
3. C.A. Ardagna, M. Cremonini, E. Damiani, S. De Capitani di Vimercati, and P. Samarati. Supporting location-based conditions in access control policies. In *Proc. of the ASIACCS'06*, Taipei, Taiwan, March 2006.
4. C.A. Ardagna, E. Damiani, S. De Capitani di Vimercati, and P. Samarati. Towards privacy-enhanced authorization policies and languages. In *Proc. of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (IFIP)*, Nathan Hale Inn, University of Connecticut, Storrs, USA, August 2005.
5. P. Ashley, S. Hada, G. Karjoth, and M. Schunter. E-p3p privacy policies and privacy authorization. In *Proc. of the ACM workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.
6. C. Bettini, S. Jajodia, X. Sean Wang, and D. Wijesekera. Provisions and obligations in policy management and security applications. In *Proc. of the 28th VLDB Conference*, Hong Kong, China, August 2002.
7. P.A. Bonatti and D. Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In *Proc. of the IEEE 6th International Workshop on Policies for Distributed Systems and Networks (POLICY 2005)*, Stockholm, Sweden, June 2005.
8. P.A. Bonatti and P. Samarati. A unified framework for regulating access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002.
9. R. Chandramouli. Privacy protection of enterprise information through inference analysis. In *IEEE 6th International Workshop on Policies for Distributed Systems and Networks (POLICY 2005)*, Stockholm, Sweden, June 2005.
10. L.F. Cranor. *Web Privacy with P3P*. O'Reilly & Associates, 2002.
11. *eXtensible Access Control Markup Language (XACML) Version 2.0*, February 2005. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

12. R. Gavriloaie, W. Nejdl, D. Olmedilla, K. Seamons, and M. Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In *Proc. of the 1st First European Semantic Web Symposium*, Heraklion, Greece, May 2004.
13. International security, trust, and privacy alliance (istpa). `http://www.istpa.org/`.
14. G. Karjoth and M. Schunter. Privacy policy model for enterprises. In *Proc. of the 15th IEEE Computer Security Foundations Workshop*, Cape Breton, Nova Scotia, Canada, June 2002.
15. OASIS. *Privacy Policy Profile of XACML*, September 2004. http://docs.oasis-open.org/xacml/access_control-xacml-2_0-privacy_profile-spec-cd-01.pdf.
16. Privacy and identity management for europe (PRIME). http://www.prime-project.eu.org/.
17. P. Samarati and S. De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, LNCS 2171. Springer-Verlag, 2001.
18. B. Thuraisingham. Privacy constraint processing in a privacy-enhanced database management system. *Data & Knowledge Engineering*, 55(2):159–188, November 2005.
19. World Wide Web Consortium. *A P3P Preference Exchange Language 1.0 (APPEL1.0)*, April 2002. http://www.w3.org/TR/P3P-preferences/.
20. World Wide Web Consortium. *The Platform for Privacy Preferences 1.1 (P3P1.1) Specification*, July 2005. http://www.w3.org/TR/2005/WD-P3P11-20050701.
21. M. Youssef, V. Atluri, and N.R. Adam. Preserving mobile customer privacy: An access control system for moving objects and customer profiles. In *Proc. of the 6th International Conference on Mobile Data Management*, Ayia Napa, Cyprus, May 2005.