Privacy and Identity Management in Europe for Life

Second research report on next generation policies

Editors:	Sabrina De Capitani di Vimercati (UNIMI)
	Pierangela Samarati (UNIMI)
Reviewers:	Uli Pinsdorf (EMIC)
	Sandra Steinbrecher (TUD)

Identifier:	D5.2.2
Type:	Deliverable
Version:	1.0
Class:	Public
Date:	February 26, 2010

Abstract

The consideration of privacy issues introduces the need for rethinking authorization policies and models and the development of novel privacy-aware paradigms for access control. Goal of Work Package 5.2 is then the definition of a flexible and comprehensive privacy-enhanced policy language, which will be used as a basis for the work performed in other work packages, especially in Work Package 5.3.

This document presents the advancements made in the research work during the second year of PrimeLife in Work Package 5.2. The document includes one chapter for each task of the work package that briefly describes the main research results along with an indication of what are the issues that will be addressed in the remaining year of the project. The last chapter lists the abstracts of the research papers reporting the findings of Work Package 5.2 published in the second year of the project.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 216483 for the project PrimeLife.

Members of the PrimeLife Consortium

1.	IBM Research GmbH	IBM	Switzerland
2.	Unabhängiges Landeszentrum für Datenschutz	ULD	Germany
3.	Technische Universität Dresden	TUD	Germany
4.	Karlstads Universitet	KAU	Sweden
5.	Università degli Studi di Milano	UNIMI	Italy
6.	Johann Wolfgang Goethe - Universität Frankfurt am Main	GUF	Germany
7.	Stichting Katholieke Universiteit Brabant	TILT	Netherlands
8.	GEIE ERCIM	W3C	France
9.	Katholieke Universiteit Leuven	K.U.Leuven	Belgium
10.	Università degli Studi di Bergamo	UNIBG	Italy
11.	Giesecke & Devrient GmbH	GD	Germany
12.	Center for Usability Research & Engineering	CURE	Austria
13.	Europäisches Microsoft Innovations Center GmbH	EMIC	Germany
14.	SAP AG	SAP	Germany
15.	Brown University	UBR	USA

List of Contributors

Contributions from several PrimeLife partners are contained in this document. The following list presents the contributors for the chapters of this deliverable.

Chapter	**Author(s)**
Executive Summary	UNIMI
Chapter 1: Policy languages (Task 5.2.1)	IBM, UNIBG, UNIMI
Chapter 2: Policy for service composition (Task 5.2.2)	EMIC, IBM
Chapter 3: Legal policy mechanisms (Task 5.2.3)	ULD
Chapter 4: Abstracts of research papers	EMIC, UNIMI, IBM, UNIBG

Executive Summary

The increased power and interconnectivity of computer systems available today provide the ability of storing and processing large amounts of data, resulting in networked information accessible from anywhere at any time. In addition, the widespread diffusion of on-line services provided by public and private organizations that require users to provide their personal information to get access to them has considerably increased the amounts of personal information collected by the service providers. This situation has led to growing concerns about the privacy of their users. Unfortunately, some of the emerging technological and organizational requirements for preserving the privacy of the users are still not completely understood; as a consequence, personal data is often poorly managed and sometimes abused.

In general, protecting privacy requires the investigation of different aspects, including the design and implementation of privacy-aware models and languages thus empowering the users and giving them the tools for effective control on personal information. The definition of privacy-enhanced policies is however complicated by the need to formally represent complex policies, where access decisions depend on the application (composition) of different rules (e.g., rules coming from laws practices and organizational regulations).

A general goal of Work Package 5.2 is the definition of a flexible and comprehensive privacy-enhanced policy language, expressing sophisticated privacy needs. In particular, Work Package 5.2 aims at providing the basis for an effective and easily deployable privacy-aware policy solution that should be adopted in current information systems with a minimal impact on the existing technologies. The findings of this work package will be directly expoited in Work Package 5.3. To pursue its objectives, Work Package 5.2 has been organized into three tasks: *Task 5.2.1: Policy languages* focuses on the definition of policy languages able to express sophisticated privacy needs; *Task 5.2.2: Policies for web service composition* focuses on the security and privacy aspects related to Web services; *Task 5.2.3: Legal policy mechanisms* focuses on the investigation of the protection models that are at the basis of current regulations.

This document presents the reseach work carried out in the context of Work Package 5.2 in the second year of the project. The work performed by the three tasks composing the work package can be summarized as follows.

- *Task 5.2.1* defined a privacy-aware language integrating cryptographic primitives and provided an extension of the XACML language and architecture for supporting actual requirements of open Web-based systems.

- *Task 5.2.2* focused on mash-ups Web services and investigated a policy language for downstream usage control.

- *Task 5.2.3* analyzed current privacy policies and legal clauses and compared them with the legal requirements.

The remainder of this document is organized in four chapters. The first three chapters (Chapters 1, 2, and 3) present the main research results obtained during the second year of the project within Tasks 5.2.1, 5.2.2, and 5.2.3, respectively, and describe the issues that will be addressed in the remaining year of the project. The last chapter (Chapter 4) lists the abstracts of the research papers reporting the findings of Work Package 5.2 published in the second year of the project.

Contents

List of Figures

List of Tables

Chapter 1

Policy languages (Task 5.2.1)

The work in this task has addressed the problem of analyzing existing languages and extending them by developing new models, languages, and policies supporting complex privacy requirements emerging in open world scenarios. This problem has been studied from different perspectives, including the possibility of exploiting recent advances in cryptography and trust negotiation that permit the use of anonymous credentials for developing privacy-aware languages and the development of a dialog management framework to enable access control interactions between clients and servers. The novel contributions described in the following leverage on the research results of PRIME (www.prime-project.eu), especially on the PRIME privacy languages [ACDS08]. The goal has been the extension of the eXtensible Access Control Markup Language (XACML) [eXt05], which is the most significant and emerging solution for controlling access in an interoperable and flexible way. In particular, we developed an access control policy language and architecture that, on one side, provide access control functionality and, on the other side, protect the privacy of the involved parties and of their personal information. The contribution of the work performed in this task is then twofold: *i)* the definition of a privacy-aware policy language and system that support cryptography for access control; *ii)* a study and definition of the extensions that need to be made to the current XACML language and architecture to include the new concepts proposed to make them easily deployable and suitable for open Web-based systems. In the remainder of this chapter, we describe these contributions in more details and outline future research.

1.1 Exploiting cryptography for privacy-aware access control

Almost everyone uses electronic means for their daily interactions with businesses, governments, colleagues, friends, and family. In these interactions, users play different roles such as customer, citizen, patient, or family member and they disclose personal information ranging from attributes such as date of birth, age, and home address to credentials pertaining to authorizations and privileges. Indeed, the number of transactions con-

ducted electronically is ever growing and in fact not limited to those over the Internet as electronic authentication and authorization with some kind of token (e.g., electronic identity cards, driver's licenses, tickets and toll-tokens) become widespread.

Organizations endeavor to control access to the resources they provide based, among other things, on the attributes of the requestor. Access control thus is one of the key functions of identity management (IdM) from the perspective of enterprizes, although it is interwoven with other functions. Identity management systems in this context maintain digital identities, or accounts, containing attributes (e.g., a name) and properties (e.g., entitlements within the system's domain such as access rights) of the entities (usually individuals) within their domain. The accounts have an identifier (e.g., username) and one or more authenticators (e.g., a password). The individual's need for control over her information is often neglected in traditional IdM systems, even though the personal information she reveals is highly sensitive.

More recently a shift in focus of identity management can be witnessed from a strict perspective of enterprize-centric access control to resources towards a perspective that takes the interests of the individual into account. A number of identity management systems are available today, being standardized, or developed that are illustrative for this change in focus. These include, for example, the open source project Higgins (`http://www.eclipse.org/higgins/`), Microsoft's CardSpace (`http://msdn.microsoft.com/en-us/library/aa480189.aspx`), and the Liberty Alliance set of protocols (`http://www.projectliberty.org/`).

Recent advances in cryptography have sparked a new generation of IdM systems based on anonymous credentials [Cha85, Bra99, CL01]. The basic concept has been known for quite some time: instead of revealing all attribute values encoded in the user's credentials, anonymous credentials allow the user to prove that she possesses valid credentials with attributes that satisfy some claim, without revealing any more information about the attributes than what is directly implied by the claim, however. More recently, more cryptographic tools have been developed that further extend the functionality of anonymous credentials, such as verifiable encryption [CS03] (i.e., a public-key encryption scheme that is compatible with an anonymous credential scheme such that it allows claims to be proved about how the encrypted content was derived from attributes in a credential - without revealing the content) and limited spending of credentials [BCC04, CHK$^+$06]. To be incorporated into IdM systems, privacy-enhancing functions need to be governed by specific privacy policies that must not only be stated, but also be enforceable by technical means. This holds for the *access control* policies (ACP), which state the conditions that a requestor must satisfy to gain access to a resource; for the *data handling* policies (DHP), which state how the requestor's information should be treated once it is revealed; as well as for the *trust policies* (TP), which state which authorities can be trusted to certify what type of information. On the other hand, the policy languages should be expressive enough to leverage the advanced features offered by the underlying technology. A policy language that combines elements from access control, data handling, and trust policies, and that is the first to model anonymous credentials and a number of related cryptographic extensions has been designed and described in [ACK$^+$10]. The proposed language has been integrated with the following cryptographic primitives.

- *Anonymous credentials.* Anonymous credentials can be thought of as digitally

signed lists of attribute-value pairs that allow the owner of such credentials to prove statements about attribute values without revealing any more information about them than what is directly implied by the statement. By issuing a credential (i.e., by signing the list of attribute-value pairs), the authority certifies that the user satisfies the described attributes. Anonymous credentials allow the user and the verifier to engage in an interactive *selective-show* protocol during which the user proves that she owns a valid credential of which the attribute values satisfy some *claim*. The only information leaked about the attribute values however is that the claim holds true.

- *Pseudonymous identification.* When a user regularly accesses the same service, she may not want to reprove at each visit that she qualifies for using the service, but may prefer to establish a permanent user account instead. To protect her privacy, she wants to do so under a pseudonym: it is bad enough that all her actions at this service now become linkable, she does not want them to become linkable to her actions across other services too. The server, on the other hand, may want to prevent users sharing their account information with others, thereby giving non-qualified users access to the service as well.

  A pseudonymous identification scheme allows a user to derive from a single master secret *msk* multiple unlinkable cryptographic pseudonyms $nym_1, nym_2, \ldots$, and later authenticate herself by proving that she knows the master secret underlying one of the cryptographic pseudonyms. Of particular interest are those pseudonymous identification schemes that are compatible with an anonymous credential scheme, allowing the same master secret key *msk* to be encoded as an attribute in *all* the credentials belonging to one user, thereby preventing sharing of credentials.

- *Verifiable encryption.* A verifiable encryption scheme [CS03] is a public-key encryption scheme that is "compatible" with an anonymous credential scheme, in the sense that it permits to prove claims about how the encrypted content was derived from attributes in a credential—without revealing the content, however.

- *Limited spending.* Certain applications require that the number of times that a credential can be shown anonymously be limited [BCC04, CHK+06]. For instance, a credential representing a wallet of n coins can be shown n times. A user can nevertheless attempt to use a credential more often. Cryptographic serial numbers[1] allow to detect overspending and, optionally, to obtain some information in escrow. The escrow is certified identifying information about the user that is hidden until an overspending occurs.

The resulting policy language uses anonymous credentials with various extensions, thus enabling system designers to take advantage of the cryptographic mechanisms to protect the users' privacy. As a matter of fact, users can carry out transactions (e.g., over the Internet) revealing only the strictly necessary personal information.

[1] A cryptographic serial number looks like a random number, but is in fact deterministically derived from a unique *seed* embedded in a credential that can generate up to the *spending limit* different serial numbers.

1.2 Extending XACML for open web-based scenarios

Open Web service systems, in which servers do not normally have any prior knowledge of users, call for a different solution from a traditional access control based on preliminary identification and authentication of requestors. The solutions proposed so far are in most cases logic-based and, although expressive, they hardly ever result applicable in practice, where simplicity, efficiency, and consistency with consolidated technology play a fundamental role [BS02, DFJS07]. Although notably widespread in the research community and the industry, and likely the most significant proposal to date, the XACML language [eXt05] still suffers from some limitations when it comes to its capabilities in supporting the requirements of open Web-based systems. Using XACML standard extension points is however possible to define new functions, data types, and policy combination methods, thus exploiting the language's flexibility to adapt it to several different needs. We proposed a profile with a specific focus on those aspects of open world scenarios that are not supported by the standard [ABD+09, ADP+09a]. The novel concepts that access control in open Web services should include have been introduced together with guidelines on the modifications that need to be applied to the current XACML language to include the proposed extensions. In the following, we describe the proposed profile for XACML that makes it easily deployable and suitable for open Web-based systems.

1.2.1 Deployment in XAMCL

XACML is not suitable for supporting the definition of access control policies in open scenarios, since it has some significant limitations. First, traditional XACML policies allow for the definition of generic boolean conditions referring to the different elements (e.g., subject, object, action) of a policy, thus enforcing an attribute-based access control. However, no support for expressing and reasoning about credentials is provided. Second, XACML neither provides the basis to reason about existing information, thus deriving new concepts, nor an infrastructure to support recursive conditions like the features offered by logic-based policy languages. Third, XACML implicitly assumes that the engine that enforces access control decisions has all the necessary information to complete the evaluation process. Such an assumption is not realistic in most cases, and we counter it by calling for an infrastructure to manage the dialog between the involved parties. We have studied how these limitations can be counteracted with the addition of novel features to XACML having a limited impact on the original specification, toward an extended XACML framework suitable for open Web services. The results of this study have been presented in [ADP+09b]. We now show how a support for certified information, abstractions, recursive reasoning, and dialog management enabling an interactive access control between clients and servers based on incremental releases of data and request for data can be deployed in XACML.

Support for credentials

Although designed to be integrated with the Security Assertion Markup Language (SAML) [AL04] for exchanging security assertions and providing protocol mechanisms, XACML lacks a real support for considering and reasoning about digital certificates,

```
<certifications>
  <certification id="IT_IC">
    <group>
      <type Disclosure="condition">identity_card</type>
      <issuer Disclosure="property">IT_Gov</issuer>
      <method Disclosure="condition">X.509</method>
    </group>
    <group>
      <type Disclosure="condition">passport</type>
      <issuer Disclosure="property">IT_Gov</issuer>
      <method Disclosure="condition">SAML</method>
    </group>
  </certification>
</certifications>
```

(a)

```
<abstractions>
    <abstraction id="id_document">
      <is>
        <item>identity_card</item>
        <item>driver_license</item>
        <item>passport</item>
      </is>
    </abstraction>
</abstractions>
```

(b)

Figure 1: XML representation of conditions on credential metadata (a) and of an abstraction (b)

in particular, for expressing conditions on certified properties, and on properties of the certificates themselves, to which we referred to as metadata (e.g., the certificate type, or its issuer). XACML currently supports attribute-based access control; we extended it to support also credential-based access control. To represent and manage credentials in XACML, and related properties and conditions on them, we have modeled conditions on metadata and conditions on the attributes separately. Attributes in the credentials have been treated as any other property and need only to be associated with the proper certification. To do that with minimal impact on XACML, we reused the *Issuer* attribute in the `SubjectAttributeDesignator` element. In particular, an occurrence of a credential attribute in the subject expression translates into an element `SubjectAttributeDesignator`, where attribute *AttributeId* is equal to the attribute name, and attribute *Issuer* refers to the metadata. A new XML schema has then been introduced to represent credential metadata. The schema has a root element `certifications` containing one or more elements `certification`. Each `certification` is composed by one or more alternative `group` elements, each containing restrictions on metadata. Element certification has an identifying attribute id whose value is the identifier with which the credential condition should be referred. Figure 1(a) illustrates an example of metadata conditions, showing a certification with attribute id equals to IT_IC that represents either a credential of type identity card issued by IT Gov with the X.509 proof method, or a credential of type passport issued by IT Gov with the SAML proof method.

Support for abstractions

Abstractions, also referred to as abbreviations, macros, or ontological reasoning in several logic-based proposals, allow for the derivation of new concepts (abstractions) from existing ones. They intuitively represent a shorthand by which a single concept is introduced to represent a more complex one (e.g., a set, a disjunction, or a conjunction of concepts). For instance, *id_ document* (abstraction head) can be defined as an abstraction for any

element in set {*identity_ card*, *driver_ license*, *passport*} of credentials (abstraction tail). A policy specifying that an access requester must provide an *id_ document* can then be satisfied by presenting any of the three credentials above. To support abstraction specification in XACML, we integrated XACML with XQuery [Boa07], a language developed by W3C for querying XML data. Abstractions have been defined via XQuery functions and referenced in XACML conditions as follows. A new XML schema defines abstractions, where a root element, called `abstractions`, includes a set of single abstractions (see an example of abstraction in Figure 1(b) showing the described before). An XQuery function, called *expansion function*, takes as input the abstraction head and produces as output the abstraction tail.

Support for recursive conditions

Recursion plays a fundamental part in the representation of restrictions on how authorities and, more in general, trusted parties delegate the ability to issue credentials. The delegation can be seen as a certification of the capability of another party to create credentials on behalf of the delegator. In distributed systems characterized by a complex architecture, delegation is a feature that increases flexibility and allows for a simple way to issue credentials, particularly in an open environment. In such systems, specifications of restrictions in delegation are needed, and the support for recursion in the policy language can be exploited to specify conditions on data with a recursive structure. As for abstractions, we proposed to use an XQuery engine to manage recursion in XACML. Again, recursive conditions have been defined via recursive XQuery functions. These functions have then been embedded and referenced in the policies, without changes to the XACML language, to define policy conditions based on recursive concepts (e.g., the supervisor concept in a business hierarchy). These functions take as input the XACML context, and produce new information to be used in policy evaluation. As a consequence, XQuery offers recursive reasoning and it allows for the creation of additional attributes to be used in the evaluation of the XACML policies during the policies evaluation process.

Support for dialog

The introduction of a dialog between the involved parties introduces several advantages, such as enabling the server to communicate which information is needed to evaluate a policy, which in turn allows the access requester to hand over only the necessary credentials, instead of her whole set, as in the current XACML proposal. The access control process thus becomes able to operate without an a-priori knowledge of the requester [ACK+10]. Extending XACML with dialog management does not only avoid the simple evaluation to indeterminate of all those cases for which the server is missing information, but it also permits to tackle the issue of the privacy trade-off between providing the whole set of credentials (on the access requester side) and disclosing the whole access control policy (on the server side). Such result has been achieved by attaching a disclosure attribute to every condition in an access control policy. Such attribute indicates what type of disclosure policy is associated with the condition, and it is then enforced by hiding from the access requester the information that cannot be released according to the specified disclosure policy. The more (less) of an access control policy is disclosed, the smaller

Basic Constructs	**XACML**
Credential metadata	⟨`certification`⟩ ... ⟨/`certification`⟩
Credential conditions	Attribute *Issuer* in element ⟨`SubjectAttributeDesignator`⟩
Abstractions	XQuery functions
Recursive conditions	XQuery functions
Dialog	Attribute *Disclosure* in any of: - element ⟨`Condition`⟩ in XACML - element ⟨`Apply`⟩ in XACML - sub-elements of ⟨`group`⟩

Table 2: From formal concepts to XACML

(bigger) is the quantity of information in terms of released credential that will have to be provided by the access requester. Differently from the extensions provided to support previous concepts, dialog management has required a change in the XACML language for representing the disclosure policies associated with conditions. Each condition appearing in a XACML policy is associated with a disclosure policy represented through a new attribute *Disclosure*. This attribute is added to those elements used for representing the conditions: elements `Condition` and `Apply` in XACML, and each sub-element of element `group` in the credential schema. The admissible values for the *Disclosure* attribute are: *i) none*, nothing can be disclosed about the condition; *ii) credential*, only the information that there is a condition imposed on some metadata/attributes can be disclosed; *iii) property*, only the information that a property needs to be evaluated can be released; *iv) predicate*, both the information that a property needs to be evaluated and the related predicate can be released; *v) condition*, all information about the condition can be disclosed.

Table 2 summarizes the mapping between the basic concepts previously described and the changes introduced in XACML.

1.3 Extended XACML architecture

Another important contribution developed in the second year of the project is the extension of the XACML architecture and modules to complement them with functionalities for an effective credential-based management and privacy support. We investigated the combination between XACML and the PRIME architecture, thus resulting in an infrastructure that provides the flexible access functionality of XACML enriched with the privacy features of PRIME. This research activity, which has been presented in [ADP+09a], has also provided architectural and implementation guidelines for the concrete realization of the model and language exploited in WP5.3.

1.3.1 Scenario and motivations

The huge success of the Web as a platform for the distribution of services and dissemination of information makes the protection of users' privacy a fundamental requirement. Privacy issues affect different aspects of today Internet transactions, and controlling access to private information plays a crucial role in providing privacy guarantees [SK07]. Although considerable work has been performed in the field of access control for distributed services [AHKS02, BS02, eXt05, W3C06], available access control mechanisms are at an early stage from a privacy protection point of view. This situation reflects the fact that attention on security requirements has been mostly focused on addressing server-side security concerns (e.g., communication confidentiality, unauthorized access to services, data integrity). We then focused on the development of a privacy-aware access control system that takes into consideration both the client and the server sides.

XACML [eXt05], the result of an OASIS standardization effort, represents today the most effective and accepted solution for controlling access in distributed environments. XACML proposes an XML-based language to express and interchange access control policies, and defines both an architecture for the evaluation of policies and a communication protocol for message exchange. Although XACML is largely adopted and is considered a reference solution, it supports neither privacy features nor the effective evaluation and specification of credential restrictions in the policies.[2] Also, it imposes that the data of the users be available at access request time, thus lacking support for negotiation/dialog between users and servers.

The consideration of privacy issues introduced the need to rethink authorization policies and models, and the development of new paradigms for access control and authorization specification and enforcement. Two major issues need to be addressed. First, access control needs to operate even when interacting parties wish to disclose limited or no information about themselves. Second, data collected/released during access control, as well as data stored by the different parties, may contain sensitive information on which privacy policies need to be applied (data handling) and should therefore be protected. These issues have been first investigated within PRIME (Privacy and Identity Management for Europe - www.prime-project.eu.org), a large-scale EU-funded research project terminated in 2008, which aimed at developing a privacy-enhancing Identity Management System that protects the personal information of the users and provides a framework that could be smoothly integrated with current architectures and online services. The goal of the PRIME project was to empower users, providing them with solutions to retain control over their personal information in interactions with other parties, also imposing constraints on subsequent data handling and secondary use. In the PRIME vision, the user should have control of personal information and negotiate its disclosure for access to a service. The result of such a negotiation is an agreement between the user and the service provider, whereby the provider collects personal data for a stated, legitimate purpose and under agreed conditions of use. The PRIME project has developed a privacy-aware access control system, which supports privacy requirements and extends traditional ac-

[2] Version 3.0 of XACML, currently (July 2009) in the preliminary "first draft" status, provides a privacy profile, which is however limited, consisting only of a few requirements and the introduction of two attributes. Credential support is limited to the evaluation of attributes issuer, time, and date associated with certificates.

cess control functionalities with support for data handling [ACDS08]. Notwithstanding the significant benefits of PRIME and of its access control implementation, PRIME has shown limited appeal to those complex business scenarios with stable legacy systems and database-centered architectures, with an already existing, well integrated, access control solution. Nowadays, the same business scenarios have shown some use of XACML for controlling access to data/resources in distributed settings, but they are reluctant to change their infrastructures to integrate the PRIME solution and its privacy functionalities.

Our work aimed at providing a solution to the above issues by integrating the XACML and the PRIME architecture. The motivation was to build a flexible framework that exploits the advantages of XACML in terms of access control and scalability, and the advantages of PRIME in terms of privacy. The integration of the XACML access control within the PRIME architecture permits to put at use the PRIME features in existing business scenarios. By reducing the amount of required changes to the business and technological infrastructure, our solution provides effective deployment of privacy-support features in current information systems. Our solution also showed the flexibility of the XACML standard and of its implementation, and can serve as a guideline for others interested in extending XACML (including PrimeLife WP5.3 whose goal is the development of a policy engine that extends XACML with the support for credentials and other privacy-related features) for capturing different protection requirements.

1.3.2 PRIME access control system

The PRIME privacy-aware access control system dealt with five main key aspects: *i) resource representation,* to specify access control requirements on resources, in terms of available *metadata* describing them; *ii) subject identity,* to specify access control conditions on the subject requesting access and its personal information; *iii) secondary use,* to allow users to define restrictions on how their information will be used and processed by external parties after its release; *iv) context representation,* to provide contextual information in a standard format for the evaluation of policy conditions; *v) ontology integration,* to exploit the Semantic Web and to allow the definition of access control rules based on generic assertions defined over concepts in the ontologies.

Figure 2 illustrates the architecture of PRIME and the flow of an access control evaluation (steps 1-6) and enforcement (steps 7-10). The architecture is composed of the following main components.

- *Decision Wrapper*: drives the access control policy evaluation and enforcement.

- *Access Control Decision Function (ACDF)*: takes access decisions for all access requests directed to resources (personal identifiable information (PII) or services). It is the core component of the PRIME access control implementation.

- *Policy Manager*: manages the overall policy life cycle by providing functions for administering policies. It also provides filtering functionality over the responses to be returned to the counterpart (user or service), to restrict the release of sensitive information related to the policy itself.

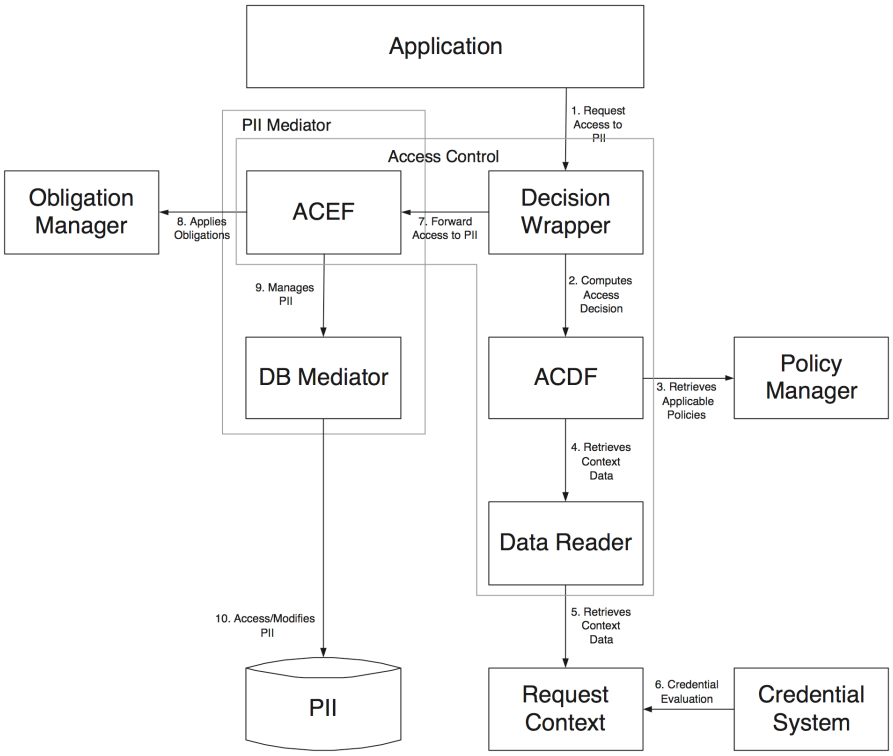


Figure 2: The PRIME access control architecture

- *Request Context*: manages all contextual information; it stores all the data and credentials released by a user in a given session.
- *Data Reader*: abstracts the communication between the ACDF and the Request Context.
- *Credential System*: responsible for credential verification.
- *Access Control Enforcement Function (ACEF)*: enforces access control decisions by mediating all accesses to resources and allowing them only if they are authorized by ACDF.
- *Obligation Manager (OM)*: responsible for managing, scheduling, enforcing, and monitoring privacy obligations. Obligations are actions that have to be performed either after an access has been granted or in the future, based on the occurrence of well defined events, e.g., time-based events or context-based events.
- *PII*: database storing all personal identifiable information.
- *DB Mediator*: abstracts the communication between the ACEF and the PII repository.

The ACDF component is responsible for taking access decisions for all access requests directed to resources, by retrieving and evaluating all access control and data handling policies applicable to a request. Access control policies govern access to and release of resources managed by a party [SD01]. Data handling policies define how data (personal information) will be (or should be) dealt with at the receiving parties [ACDS08, W3C06]. An access request is modeled as a 4-uple of the form ⟨*subject, action, object, purpose*⟩, where *subject* is the optional identifier/pseudonym of the requester, *action* is the action that is being requested, *object* is the resource on which the requester wishes to perform the action, and *purpose* is the purpose (or a set of purposes) for which the access is requested. The ACDF produces the final response possibly combining the access decisions coming from the evaluation of different policies. The ACDF can then return three different decisions: *i) Yes*, the request can be granted; *ii) No*, the request must be denied; *iii) Undefined*, current information is not sufficient to determine whether the request can be granted or denied. In this case, additional information is needed and the counterpart will be asked to provide such an information.

As illustrated in Figure 2, the ACDF mainly interacts with the Request Context component. This component keeps track of all contextual information, combines information from various context sources, and describes new contextual information from this aggregation. Note that the communication between ACDF and Request Context components is mediated by a façade component, called Data Reader. The Data Reader component abstracts the process of retrieving the information needed by the ACDF for the evaluation. This approach adds a level of isolation that guarantees a simple integration of the ACDF with different context formats or modules. The Request Context component also interacts with the Credential System, which is a credential verification component, in charge of verifying (possibly anonymous) credentials.

The evaluation flow of the ACDF is as follows. After receiving the access request, the ACDF: *i)* retrieves the access control policies by querying the Policy Manager (PM), *ii)* evaluates the access control policies, *iii)* collects the data handling policies attached to the object of the request, *iv)* evaluates the data handling policies, *v)* generates a single access decision.

The ACDF supports conditions to be evaluated on both certified data, issued and signed by authorities trusted for making the statement (credentials), and uncertified data, possibly signed by the data owner (declarations). Credentials and declarations relevant to an evaluation process are retrieved from the Request Context component. In case of credentials, the Request Context component retrieves the information needed by the ACDF by using the Credential System component.

1.3.3 The PrimeLife access control system

PRIME and XACML represented starting points for PrimeLife, and their careful integration has provided important functionalities that neither XACML nor PRIME alone can provide. XACML, in fact, represents the most accepted and flexible access control language, but it does not provide effective support for privacy. PRIME, instead, provides privacy functionalities, but its impact on real world is limited by the fact that PRIME adoption in existing businesses would require important changes to legacy systems. Taking into account this analysis, we decided to integrate the XACML evaluation and en-

forcement features with the PRIME privacy functionalities. By integrating XACML into the PRIME architecture, we have all the advantages of the XACML-based solution (e.g., extendability, flexibility), enriched with support for credentials, dialog between parties, incremental release of data (e.g., interactive enforcement), and data handling. Such an integration however requires a limited change of XACML that needs to be widely accepted for making the proposed solution usable in practice. To enable coexistence of PRIME and XACML, the PrimeLife architecture has employed two independent modules that have separate duties. For the sake of clarity, we use PrimeLife XACML Engine to denote the enhanced XACML Engine, and Data Handling Decision Function (DHDF) Engine to denote the modified version of the PRIME ACDF (see Figure 2). Specifically, the PrimeLife XACML Engine is devoted to the evaluation and enforcement of access control, while DHDF provides privacy and data handling functionalities. Differently from ACDF in the PRIME solution, DHDF does not implement a complete privacy-aware access control system, but rather it is responsible for the management and evaluation of data handling policies only.

The integration exploits a mechanism that the PrimeLife XACML architecture can use to freely access any external repository via a refined Policy Information Point (PIP) component. The context information stored in the Request Context, and produced during the interaction with the requester, is an example of such repositories. PrimeLife XACML can then access the Request Context, to evaluate its policies, by means of a standard PIP extension, and can use the PRIME architecture to negotiate access and evaluate credentials.

Figure 3 depicts the resulting PrimeLife architecture. When an access control decision needs to be taken, the Decision Wrapper component forwards the access request to the AC Manager, together with the possible relevant data handling policies attached to the requested resources. The AC Manager forwards the access request to both the PrimeLife XACML Engine and the DHDF Engine and then takes the final decision by combining their answers. The DHDF Engine receives the request with the associated data handling policies, while the PrimeLife XACML Engine accesses the Policy Manager to retrieve the applicable access control policies. Finally, both the DHDF and the PrimeLife XACML engines access the Request Context via the Data Reader component and retrieve the information (possibly certified by the Credential System) needed for the evaluation.

Figure 4 shows a detailed representation of the data flow of the PrimeLife XACML Engine in the PrimeLife architecture (Figure 3). The structure presents a strong correspondence with the standard XACML architecture, and traditional XACML components can be easily mapped onto the modules. The data flow is as follows. First, the access request is forwarded by the AC Manager to the PrimeLife Policy Enforcement Point (PEP) (step 1), which is an extended version of the standard XACML PEP. The PrimeLife PEP is responsible for managing the interaction with the AC Manager, providing functionality for supporting negotiation (i.e., request for additional data). The PrimeLife PEP component creates an XACML request and forwards it to the Context Handler (step 2). The Context Handler, as in the standard XACML architecture, invokes the PrimeLife Policy Decision Point (PDP) (step 3). The PrimeLife PDP is responsible for evaluating traditional XACML policies and extends the standard XACML PDP component to access the Policy Manager. The PrimeLife PDP then accesses the Policy Manager to retrieve the XACML policies applicable to the request (step 4). To this aim, the Pol-

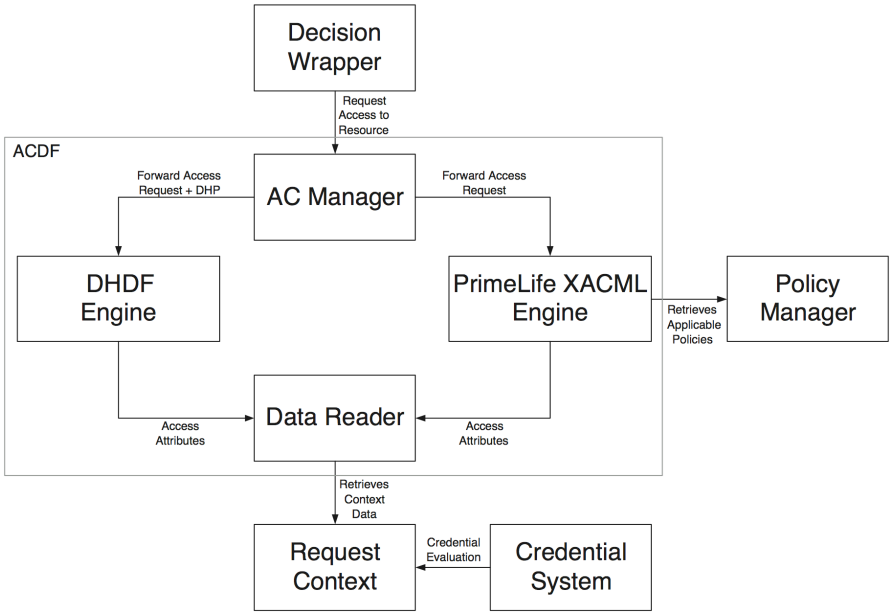


Figure 3: The PrimeLife access control architecture

icy Manager provides an external interface compliant with a standard PAP component. After collecting the applicable policies, the PrimeLife PDP accesses the Context Handler to retrieve the data needed for the evaluation (step 5). If some data necessary for taking a decision are missing, the Context Handler may access different PIP extensions. These extensions include an enhanced PIP (PrimeLife PIP) that communicates with the Data Reader to access the Request Context (steps 6-9). After receiving all data, the PrimeLife PDP (step 10) finally evaluates the applicable policies and takes a decision. The final decision is sent to the Context Handler (step 11) and then forwarded to the PrimeLife PEP component (step 12). Finally, the access decision is returned to the AC Manager (step 13), which combines it with the answer coming from the evaluation of data handling policies performed by the DHDF.

The main advantages achieved by the proposed architecture with respect to the standard XACML implementation are as follows.

- *Negotiation (minimal disclosure and interactive enforcement).* The current XACML architecture does not support negotiation between the parties. An XACML access decision can assume four values (i.e., *permit*, *deny*, *indeterminate*, *not applicable*). Indeterminate is returned if no decision can be taken since some information needed to evaluate the policy is missing. For instance, an access is restricted to users of a certain country, but the nationality of the requester is unknown. If the decision returns indeterminate, according to a safe-default approach, XACML simply denies the access. Our solution enhanced XACML with the abil-

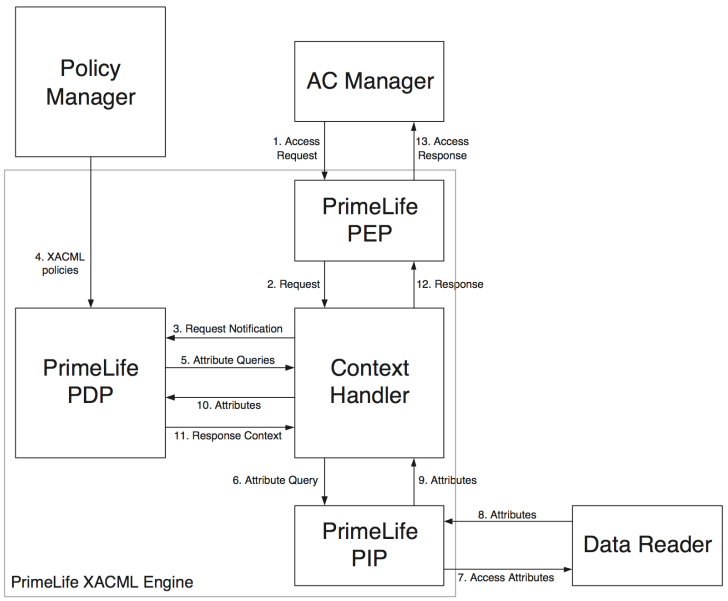


Figure 4: The PrimeLife XACML Engine data flow

ity of dialoguing with the user to communicate the fact that certain information needed for evaluating the access control policy is missing, allowing then the user to provide it and, possibly, successfully acquire access. The Decision Wrapper is responsible for the dialogue between the server and the user that consists in communicating all the conditions that have to be satisfied. At each access control request, if the policy evaluation results indeterminate, the AC Manager returns to the user the attributes that need to be evaluated for taking the access decision. The user can then decide to provide the requested attributes. Internally, the processing of the response to the user works as follows. When the PrimeLife PDP component generates an *indeterminate* response, it returns such a response to the PrimeLife PEP together with the information on which attributes are needed for a decision to be taken. This communication is enforced via a side channel provided by the PrimeLife PIP, and accessible by the PrimeLife PEP component. The use of a side channel is necessary to be able to communicate attributes, since the response format used by the standard XACML PDP engine can assume only the allowed four values and cannot represent that information. At evaluation time, the PrimeLife PIP component has the responsibility of identifying the attributes that are unknown and need to be evaluated for taking the access control decision. PrimeLife support for negotiation allows the client and the server to incrementally exchange data and request for data preserving, on one side, the principle of the *minimum disclosure* and, on the other side, supporting an incremental evaluation of the policies.

- *Credential-based restrictions.* While XACML acknowledges that properties can be presented by means of certificates, it does not provide a real support for expressing

and reasoning about digital credentials in the specification and evaluation of the authorization policies. By contrast, PRIME supports requests for credentials and certified attributes in the policies. Also, PRIME can reason about credentials. For instance, PRIME supports the definition of policies requesting information (i.e., a set of attributes) to be certified by the same credential. Credential information is stored in the Request Context and is accessible by the PrimeLife XACML Engine. The XACML architecture has then been enhanced to support credential-based restrictions, called *evidences* in the PRIME architecture, without requiring changes to the language schema. Rather, the XACML language has been enhanced with credential specification by changing the semantics of the *issuer* attribute, originally defining the issuer of a specific attribute. In our solution, the *issuer* attribute is used as a reference to a specific evidence, stored as a separate XML file and containing the restrictions on credentials, to be evaluated on credential metadata (e.g., issuer, type). The PrimeLife XACML Engine (see Figure 4) evaluates credential-based restrictions as follows: *i)* the PrimeLife PDP component loads a policy, together with information related to the evidences; *ii)* through the PrimeLife PIP component, the PrimeLife PDP accesses information needed to evaluate the evidences of the policies and stored in the Request Context; *iii)* the PrimeLife PDP evaluates the policies and the evidence conditions on the credential metadata.

- *User-driven constraints.* While XACML is designed to manage access control only, and is focused on service side restrictions, the system illustrated in this section takes a different approach and aims at providing the users with a solution for protecting their privacy. Our solution, in fact, in addition to policies and mechanism to manage and evaluate traditional access control policies, provides a fully compatible and extensible solution that supports the definition, evaluation, and enforcement of privacy policies defined by the users themselves. These privacy policies (i.e., data handling policies) are defined through an ad-hoc syntax and are evaluated together with the XACML policies to provide a full-fledged privacy-aware access control system, that permits users to protect their own data.

1.4 Future research

The future work for Task 5.2.1 will continue the investigation along the lines of work previously presented. In particular, we plan to investigate the following three different, but related, research directions.

- *Anonymous credentials.* The extensions we proposed for XACML and presented in [ADP+09b] assume that users present their certificates to the server that are then used to verify whether the user satisfies specific conditions defined on the certified attributes. It would then be interesting to extend this credential-based access control to accommodate the presence and management of *anonymous credentials*, which allow users to prove that they satisfy given conditions about attribute values without however revealing the values of such attributes.

- *Policy communication.* Credential-based access control implicitly requires that the server should be able to formulate a credential request basically saying what are

the conditions that a user should satisfy to gain an access. A naive way to formulate a credential request consists in giving the user a list with all the possible sets of credentials that would enable the service. This solution is clearly not feasible, due to the large number of possible alternatives. It is also not possible to release the conditions that a user should satisfy since they may include sensitive information. Although different proposals have presented sophisticated dialog strategies for policy communication where there is a bidirectional exchange of request-response messages, these approaches are not suitable for a wide range of use cases that request low complexity, low overhead, and high performance. It would then be interesting to define novel dialog solutions that should take into consideration the fact that often the conditions specified in a policy cannot be communicated to the counterpart as they are.

- *Selective release of certified attributes.* Credentials are usually considered as atomic objects whose certified attributes cannot be selectively extracted. Anonymous credentials allow instead the possibility of "extracting" single properties. It would then be interesting to investigate how this possibility may affect a credential-based access control and how can be exploited for the goal of minimizing the information released during the evaluation of an access request.

Chapter 2

Policies for service composition (Task 5.2.2)

Many web services today are so-called service *mash-ups*. A mash-up is a service that acts as a front-end for a composition of multiple subservices that run on different hosts and are offered by different companies. For example, a travel booking mash-up may offer an integrated interface to book flights, hotels, and rental cars. In the background, however, it invokes the web service APIs of different specialized airline, hotel, and car rental subsidiaries to collect offers. The best offers are presented to the user, who selects an offer, enters her booking and payment information, and confirms to let the mash-up make all the bookings for her through the subsidiaries' APIs.

Service mash-ups are important for leveraging online service APIs to create new functionality, but pose significant privacy risks for their users. The users cannot keep track of who stores what information about them, and often they do not even know the subsidiaries their data is shared with.

To overcome this problem, service providers publish their privacy policies to inform the users about how the gathered data is used. Human-readable privacy policies have the obvious disadvantages of being written in a complex "legalese" language and mostly being ignored by users. Even if clear privacy policies are presented, they often remain vague about the sharing of information with third parties. For example, the privacy policy of Expedia.com[1], a popular online travel service, mentions the following with regard to sharing personal information with suppliers:

> *We do not place limitations on our suppliers' use or disclosure of your other personal information* [i.e., other than the email address]. *Therefore, we encourage you to review the privacy policies of any travel supplier whose products you purchase through this site.*

Machine-interpretable privacy policy languages such as EPAL [AHK+03] and P3P [W3C06] are a very promising approach, in particular when used in combination

[1] cf. `http://www.expedia.com/daily/service/privacy.asp`

with a privacy preferences language such as APPEL [W3C02]. In the latter language, users can express how they expect their data to be treated, so that an automated or semi-automated matching procedure can decide on the acceptability of a proposed P3P policy.

Unfortunately, EPAL and P3P are both rather constrained in expressivity regarding sharing personal information with third parties, or *downstream usage* as we call it here. EPAL leaves the definition of specific actions and obligations up to enterprise-defined vocabularies, and is hence silent about downstream usage. Support in P3P is limited to specifying which of six classes of third-party recipients the information will be shared with; it is up to the server to classify his recipients into one or more of the classes. SecPAL for Privacy (S4P) [BMB09] proposes a logic-based language to specify human readable policies and preferences. S4P does not focus on downstream usage control and does not specify downstream in terms of access control.

The abstract scenario that we considered in this task is the following. Two parties, typically a user and a server, engage in an interaction where one of the parties, typically the server, requests some personally identifiable information (PII) from the other party. We will from now on call the party that provides the data the *data provider* and the party that requests the data the *data consumer*. Moreover, we consider a scenario where at a later point in time, the data consumer may want to forward the PII to a third party, called the *downstream data consumer*.

In this task, we have investigated how one would structure a policy language for downstream usage control. The difficulty here is that downstream usage control involves a mixture of what is typically considered access control (who is allowed to receive the data, e.g., by roles or owned credentials) and data handling (how is the recipient supposed to treat the data, e.g., usage purposes or retention period). We consider the most general setting here, where the provider states in her privacy preferences, for each hop in the chain of downstream consumers, how they have to treat her data and to whom they can further forward it. At the same time, each consumer specifies in his privacy policy how he intends to treat the data and to whom he intends to forward it. We propose XML-based languages to express both the provider's preferences and the consumers' policies, and describe an automated matching algorithm to determine whether the proposed policies are allowed by the specified preferences. The provider and the consumers can specify downstream usage restrictions up to any number of hops (not necessarily the same number). Optionally, they can either specify that the last restrictions in the chain are valid for all subsequent hops, or that after that hop no further downstream usage is allowed. Moreover, for situations where the consumer does not know at the time of data collection to whom he may forward it, our language allows the server to perform "lazy" matching, in the sense that he declares to be willing to impose any restrictions on further downstream usage that the provider may specify.

2.1 Related work and contributions

Our work is closely related to rights expression languages (RELs), privacy policy languages, and usage control. We give a brief overview of each of these lines of work and how they relate to the language we propose.

2.1.1 Rights expression languages

From a *protocol* point of view, there is a clear difference between privacy policies on the one hand, and digital right management (DRM) and enterprize right management (ERM) on the other hand. Indeed, in privacy, it is usually the consumer (data controller) who imposes the policy, while in right management, the author (or publisher) imposes the policy (license) to the consumer. The main purpose of *matching* is to help the end-user deciding whether he accepts the service and the policy. Such matching is interesting to make decisions related to privacy or to rights management and, in both case, is generally done by the user.

From a *trust model* point of view, privacy policies and enterprize right management are similar because they assume "honest consumers", data controller and consumer respectively, which are willing to enforce the policy (accept audit, use required client application, etc.). In digital right management, the threat model is different since consumers may mount attacks in order to violate the policy. As a result, trusted hardware and/or software are required in DRM.

There are three key differences between our approach and state of the art right expression languages used in ERM and DRM, e.g., MPEG-21 REL [Wan04], XrML [Con02], and ODRL [ODR02]:

- RELs focus on rights (e.g. print, play) and add some conditions and constrains (temporal, fees, device, etc.) but obligations remain underspecified.
- Matching is generally not specified even if automating the process of accepting an offer, i.e. a proposed license, would make sense.
- Downstream data sharing (transfer, i.e. sell, give, lease) is also defined as a right. However, it is not as expressive as our model and does not result in a matching algorithm.

The most important difference between right management and our work is the fact that in the former the data provider pushes sticky policies (i.e., licenses) onto the consumer without matching. In a privacy setting, the provider (i.e., user) usually does not have this power. Rather, it is the consumer who imposes a policy onto the provider.

Another major difference is that the domain-specific vocabulary is quite different for RELs and for privacy policies, even though overlaps exist, e.g., the obligation to delete data within a certain time makes sense in both domains. Apart from the vocabulary though, the same overall language structure should be usable for both.

2.1.2 Privacy policy languages

We already briefly discussed the shortcomings of the privacy policy languages EPAL [AHK+03], P3P [W3C06], and APPEL [W3C02] in terms of expressing restrictions on the downstream usage. EPAL is mainly intended for writing enterprize-internal privacy policies to govern data handling practices. P3P, on the other hand, is mainly intended for websites to communicate their privacy practices to the outside world. APPEL is a preference language for P3P, i.e., enabling users to specify their privacy preferences and automatically match those against proposed P3P policies.

SecPAL for Privacy (S4P) [BMB09] is a logic-based language to specify privacy policies and preferences. S4P specifies preferences as `may` assertions (i.e. authorizations) and `will` queries (i.e. obligation request). S4P specifies policies as `will` assertions (i.e. commitment on obligations) and `may` queries (i.e. authorization request). In S4P, matching is about evaluating queries with a set of assertions while, in the work presented in this document, matching is done by comparing statements.

Ardagna et al. [ACDS08] describe a data handling policy language that allows for specifying data recipients, usage purposes and obligations. In the proposed model, service providers present policy templates to users that the users may customize on the base of their own data handling policies. The outcome of a successful customization is a policy traveling with the data. The customization process is described as potentially automated, it remains unclear though how this automation can be achieved. In particular, because neither an obligation vocabulary nor matching semantics for purposes and obligations are defined. The authors do not distinguish between rights and obligations. Their *purpose* can be seen as dedicated right, however, there is no explicit right to share data downstream. The data recipient concept allows for specifying global rules on who may receive the data, and could therefore be seen as abstract downstream right. However, transitive downstream data disclosures are not explicitly discussed and specific paths along which data may be shared can not be expressed.

2.1.3 Usage control

Usage control [ABP09, HBP05, PS04] is a generalization of access control that is also concerned with how data is treated *after* it has been given away. In contrast, pure access control only addresses how data is protected *before* it is released.

Usage control takes place in a distributed setting where a process acting in the role of a *data provider* sends sensitive data to a process acting in the role of a *data consumer* based on provisions and obligations. Access and usage control requirements are stated in *usage control policies*. For expressing such policies, a number of specification languages have been developed, however, the policies of Leumann [Leu07] come closest to our approach. They are expressed by a number of rules whereby a *rule* specifies its applicability, provisional actions and obligations. They additionally contain *optional actions* and *contracts*, which reflect the obligations, and are expressed in XACML.

The concept of policy evolution for distributed usage control as proposed by Pretschner et al. [PSSW08] bears similarities to our concept of downstream usage control. The Obligation Specification Language (OSL) proposed in [PSSW08] allows the data provider to specify which consumers (indicated by their roles) have to adhere to which rights and duties when receiving the data. The language we propose differs from OSL in two important aspects. First, we envision a double-sided setting where both the data consumer and the data provider have policies (resp., preferences) describing how they will treat the data (resp., want the data to be treated). These policies and preferences are then automatically matched to yield a sticky policy that acts as the agreed-upon contract. In the OSL in contrast, it is the data provider who unilaterally describes the sticky policy that has to be adhered to, which we think is especially unrealistic in privacy, where the data provider is a private user. Second, our language can describe (and automatically match) the full path that the data is allowed to follow,

including who is allowed to share the data with whom, and how many hops the data is allowed to take. In contrast, in OSL one can specify which role of consumers have to adhere to which usage control policy. For example, in our language a patient could impose one usage control policy when a health insurance company obtains her medical record through the hospital, and another policy when it obtains the record from the patient directly. In OSL both cases are treated the same.

2.2 The language

Both the data provider and the data consumer have their own policies expressing the required, respectively the proposed, treatment of the PII. These policies contain access control and usage control policies. Access control policies state conditions that have to be fulfilled *before* a requested piece of PII is transmitted to the data consumer. Usage control policies state how the sent resource has to be treated *after* it has been transmitted. A piece of PII is only sent to a data consumer after (1) the access control requirements have been met, and (2) a suitable usage control policy has been agreed upon. We distinguish between three kinds of policies.

Preferences: In his *preferences* the data provider describes, for specific pieces of PII, which access control requirements a data consumer has to satisfy to obtain the PII, as well as the usage control requirements according to which the PII should be treated after transmission. These requirements may include *downstream usage requirements*, meaning the requirements that a downstream data consumer has to fulfil to obtain the PII from the (primary) data consumer.

Policy: The *policy* is the data consumer's counterpart of the data controller's preferences. In his policy the data consumer contains, for specific pieces of PII to be obtained, his certified properties (roles, certificates, etc.) that can be used to fulfil access control requirements, and a usage control policy describing what he is planning to do to the PII.

Sticky policy: The *sticky policy* describes the mutual agreement concerning the usage of a transmitted piece of PII. This agreement is the result of a matching process between a data providers' preferences and a data consumer's policy. (If no match is found, the data transfer should be canceled, unless the provider chooses to overrule her own preferences.) Technically a sticky policy is quite similar to preferences as described above, but it describes a mutual agreement between the data provider and the data consumer that cannot be changed. After receiving the PII, the data consumer is responsible for storing and enforcing the sticky policy.

Preferences model

To illustrate our ideas we describe a simple XML-based language to express preferences, policies, and sticky policies. In Figure 5 we give an example of Alice's preferences where she specifies that she's willing to reveal any of her email addresses to book shops, who can use it for statistics, contact, and account administration, and who have to delete

```
<Preferences>
  <Preference>
    <Applicability>
      <DataType> Address </DataType>
    </Applicability>
    <ACUC id="ACUCbookshop@alice">
      <AccessControl>
        <Rule>CertifiedAsBy{role=bookshop, issuer="CAx"}</Rule>
      </AccessControl>
      <UsageControl>
        <Rights>
          <UseDownstream allowLazy="false">
            <ACUC xsi:type="ACUCContentType" id="ACUCshipping@alice">
              <AccessControl>
                <Rule>CertifiedAsBy{role=shipping, issuer="CAy"}</Rule>
              </AccessControl>
              <UsageControl>
                <Rights>
                  <UseDownstream allowLazy="false">
                    <ACUC reference="ACUCshipping@alice"/>
                  </UseDownstream>
                  <UseForPurpose>statistics</UseForPurpose>
                  <UseForPurpose>shipping</UseForPurpose>
                </Rights>
                <Obligations>
                  <DeleteWithin>P7D</DeleteWithin>
                </Obligations>
              </UsageControl>
            </ACUC>
          </UseDownstream>
          <UseForPurpose>statistics</UseForPurpose>
          <UseForPurpose>contact</UseForPurpose>
          <UseForPurpose>accountadmin</UseForPurpose>
        </Rights>
        <Obligations>
          <DeleteWithin>P2Y</DeleteWithin>
        </Obligations>
      </UsageControl>
    </ACUC>
  </Preference>
</Preferences>
```

Figure 5: Example of preferences

it within two years. Book shops can forward it to shipping companies under the same preferences as specified above. The shop has to delete the email address within 7 days.

The `Preferences` root element contains multiple `Preference` elements, each describing to which PII it applies and what the respective access and usage controls are.

A `Preference` can refer to the applicable PII by their data type, meaning that the `Preference` applies to all PII of this type, or by a unique identifier pointing to a single instance of PII. A full-blown language would probably offer more powerful mechanisms to specify applicability, allowing for example attribute expressions or temporal constraints. If multiple `Preference` elements apply to a single piece of PII, then satisfying the conditions in either of them results in a match. In other words, `Preference`s are combined according to "or" semantics: a match occurs if the first preference matches, or if the second matches, etc. This makes it possible to define more permissive exceptions to general

preferences.

Within a `Preference`, access and usage control requirements occur in a pair enclosed in an `ACUC` element. A pair (AC, UC) means that any data consumer satisfying access requirements AC can obtain the PII when adhering to usage requirements UC. To allow multiple AC/UC combinations for a single piece of PII, one can use multiple `Preference` elements with the same `Applicability`.

We assume a simple role-based model for access control here; again, we imagine that a full-blown language would be more expressive, but we focus on the structure of downstream usage restrictions here.

Usage control requirements are expressed as a pair of `Rights` and `Obligations`. A right states an action that the data consumer is allowed to perform on the data, but doesn't have to perform to comply with the policy. An obligation states an action that a data consumer is obliged to perform. We only model two types of rights and obligations here, being `UseDownstream`, `UseForPurpose`, `DeleteWithin`, and `NotifyOnAccess`. Rights and obligations within a `UsageControl` element are combined by "and" semantics, meaning that the data consumer obtains all the specified rights and has to adhere to all of the specified obligations.

Policy model

A data consumer's policy states which usage control requirements it is willing to adhere to when requesting a specific resource from a data provider. In addition it states what properties it can prove about itself to fulfill the data provider's access control requirements. A data consumer's policy follows a similar structure as the preferences.

Downstream usage

The crucial aspect of our policy language is that it allows both the data provider and the data consumer to express to whom and under what conditions the PII can or will be forwarded. These conditions are expressed in `UseDownstream` elements.

Each `UseDownstream` element contains exactly one `ACUC` child element. Either this `ACUC` element contains a fully specified pair of access and usage control requirements, or it contains a `reference` to another `ACUC` element.

In the first case, the access control requirements explicitly specify to which downstream data consumers the PII can be forwarded, while the usage control requirements specify how these downstream consumers are supposed to treat it. Of course, the usage requirements can on their turn contain `UseDownstream` elements that specify to whom and under what conditions the downstream consumer can further forward the PII, which on their turn can contain `UseDownstream` elements, etc. This mechanism enables the data provider to restrict the forwarding of his PII up to an arbitrary number of hops. We refer to this approach as *nested* downstream usage control.

Of course, to specify preferences as detailed in Figure 5 for an entire chain of downstream consumers, the data provider must have a very good understanding of the consumers' business process. More typically, the access control restrictions in the preferences will contain more generic restrictions, such as the consumer being approved by the Better Business Bureau.

While in the second case we could in principle allow the `reference` attribute to point to any `ACUC` element, for the sake of manageability we insist that it can only refer to its closest ancestor `ACUC` element, i.e., its ancestor four levels higher in the XML tree. This essentially means that the data consumer can further forward the PII under the same restrictions that were imposed on himself. We therefore call this approach *recursive* usage control. Note that our policy language allows to combine nested and recursive usage control by defining a chain of nested usage controls for the first number of "hops" and a final recursive usage control for any further hops.

Each `UseDownstream` element can contain *at most* one `ACUC` child element, but one `Rights` element can contain multiple `UseDownstream` elements.

In many situations, the downstream consumer or his policy may not be known yet at the time the PII is transmitted to the primary consumer. Rather than specifying all intended hops in full detail, the data consumer can indicate that he is willing to enforce any restrictions imposed by the data provider by setting the `allowLazy` attribute of the `UseDownstream` element to `true` and omitting the `ACUC` element. The matching between the data provider's preferences and the downstream consumer's policy is then done by the primary consumer at the time that the PII is forwarded to the downstream consumer. In the next section we give more details on lazy matching. When occurring in `Preferences`, the `allowLazy` attribute indicates whether the data provider is fine with lazy matching being used for the restrictions expressed in the child `ACUC` element.

Finally, the `maxDownstreamDCs` attribute can be set to indicate to any integer or the value `unbounded` to indicate how many different downstream consumers the PII can maximally be forwarded. This limit refers only to the number of direct downstream consumers, and not the total number of downstream consumers after multiple hops, as this may be hard to enforce.

2.3 Matching

Given a data provider's preferences and a consumer's policies, matching aims at automating the process of deciding whether the provider can safely transmit a piece of personal data. We do so by defining a *more permissive than* operator on preferences and policies; we say that there is a match when the data provider's preferences are more or equally permissive than than the data consumers' policies.

2.3.1 Matching privacy preferences and policies

To explain the matching procedure, we use a set-based representation of the XML structure described in the previous section. A `Preferences` element is represented by a set $Prefs$ containing an element $Pref \in Prefs$ for each of its `Preference` child elements in the XML structure. For each $Pref \in Prefs$, $Pref.App$ and $Pref.ACUC$ represent the contained `Applicability` and `ACUC` child elements, respectively. The set $ACUC.AC$ is the set of access control rules contained in the `Rule` elements of the embedded `AccessControl`, while $ACUC.UC$ is the set of usage control in terms of rights ($UC.Rights$) and obligations ($UC.Obls$), specified in the `Rights` and `Obligations` elements, respectively. We use an analogous notation for the consumers' policies.

Intuitively, preferences *Prefs* are more (or equally) permissive than policies *Pols*, denoted $Prefs \trianglerighteq Pols$, if the access control properties in *Pols* satisfy the rules in *Prefs* and if *Pols* asks for less rights and promises to adhere to stricter obligations than specified in *Prefs*. We "overload" the notation of the operator $\trianglerighteq$ by using it to compare not only preference and policy sets, but also individual rights and obligations, as well as access and usage control policies.
We describe how the matching of policies and preferences boils down to the matching of individual rights and authorizations. The first rule says that preferences and policies match when for each ACUC pair in a policy there exists a matching, more permissive ACUC pair in a preference:

$$\begin{aligned} Prefs \trianglerighteq Pols \;\Leftrightarrow\; & \forall Pol \in Pols\ \exists Pref \in Prefs\ : \\ & (Pref.App \supseteq Pol.App) \;\wedge\; (Pref.ACUC \trianglerighteq Pol.ACUC) \end{aligned} \tag{2.1}$$

Pairs of access control and usage control policies are matched as follows. Note that (2.2) is evaluated multiple times during the evaluation of (2.1). For instance, $ACUC_{Pol}$ is instantiated subsequently with $Pol_i.ACUC$ for all Pol_i in *Pols*.

$$\begin{aligned} ACUC_{Pref} \trianglerighteq ACUC_{Pol} \;\Leftrightarrow\; & ACUC_{Pref}.AC \trianglerighteq ACUC_{Pol}.AC \;\wedge \\ & ACUC_{Pref}.UC \trianglerighteq ACUC_{Pol}.UC \end{aligned} \tag{2.2}$$

Determining whether an access control policy is more permissive than another one is done by checking that for each `Rule` in the preferences there is a corresponding `Property` in the policy. This could be extended to cover more complex claim-based access control and hierarchical roles.

$$AC_{Pref} \trianglerighteq AC_{Pol} \;\Leftrightarrow\; \forall r \in AC_{Pref} : r \in AC_{Pol} \tag{2.3}$$

Usage control is matched as follows:

$$\begin{aligned} & UC_{Pref} \trianglerighteq UC_{Pol} \;\Leftrightarrow \\ & \quad \forall R \in UC_{Pol}.Rights : (\exists R' \in UC_{Pref}.Rights : R' \trianglerighteq R) \;\wedge \\ & \quad \forall O \in UC_{Pref}.Obls : (\exists O' \in UC_{Pol}.Obls : O \trianglerighteq O') \end{aligned} \tag{2.4}$$

The effect of the more permissive than operator on individual rights and obligations has to be specified per right and obligation type. As examples, we specify the matching of two obligations and two rights (more complex examples are available in [Rag09]). If obligations R and R' specify that the user must be notified when her data is used, $R' \trianglerighteq R$ is evaluated with appropriated $\trianglerighteq$ operator, i.e. (2.6).

Letting obligation $DelWithin(t)$ denote a `DeleteWithin` element with duration t, we have that:

$$DelWithin(t) \trianglerighteq DelWithin(t') \;\Leftrightarrow\; t \geq t' \ . \tag{2.5}$$

Letting obligation $NotifyOnAcc(c)$ denote a `NotifyOnAccess` element with contact information c and letting ϵ denote the empty string, we have that:

$$NotifyOnAcc(c) \trianglerighteq NotifyOnAcc(c') \;\Leftrightarrow\; c' = \epsilon \;\vee\; c = c' \ . \tag{2.6}$$

Letting right $UseForPurp(p)$ denote a `UseForPurpose` element with purpose p, we have that:

$$UseForPurp(p) \trianglerighteq UseForPurp(p') \;\Leftrightarrow\; p = p' \tag{2.7}$$

For the sake of readability, support for hierarchical purposes is not describe here.

2.3.2 Proactive matching of downstream rights

We have left to define the matching procedure for downstream usage right. Supporting nested and recursive ACUC (access control and usage control) has an impact on matching. This section provides the intuition behind proactively matching a downstream structure, i.e., matching structures where both the preferences and the full chain of downstream usage policies are known at the time of matching.

For a given pair $ACUC$, let $|ACUC|$ be the "local" ACUC, meaning containing only those restrictions and obligations that do not affect downstream usage, meaning

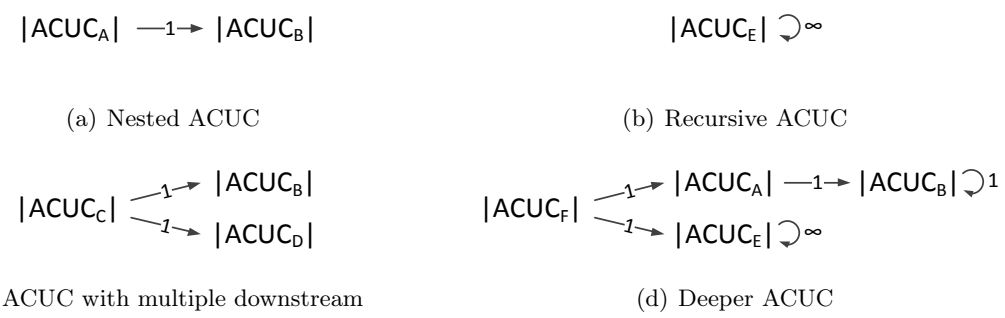
$$\begin{aligned} |ACUC|.AC &= ACUC.AC \\ |ACUC|.UC.Obls &= ACUC.UC.Obls \\ |ACUC|.UC.Rights &= \{R \in ACUC.UC.Rights \;:\; R \neq UseDS(\cdot,\cdot)\} \end{aligned}$$

We define the right of sharing downstream as $UseDS(\cdot,\cdot)$. Using this notation, we can represent the structure of an ACUC policy with downstream usage as a directed graph where each node represents a hop in the downstream usage. Each node is labeled with the local ACUC policy describing how the data are to be treated locally. Each edge represents the permission (in case of a provider's preferences) or intention (in case of a consumer's policy) to forward the data under the restrictions specified by the ACUC of the endpoint of the edge. Edges are annotated with the depth d of the recursion, where $d = 1$ when nested and $d \in \{1 \cdots \infty\}$ when recursive. For instance, the case where $ACUC_A$ permits the right to share downstream under $ACUC_B$, but prohibits any further forwarding is depicted in Figure 6(a).

By the restrictions that we imposed on the connection among ACUC pairs, the structure of the graph is roughly that of a tree where the leaf nodes can optionally have a loop, representing recursion in the downstream usage policy. Figure 6(b) for example represents a simple recursive ACUC. Figure 6(c) is an example with two downstream ACUC policies. Figure 6(d) shows a deeper nested structure.

Intuitively, matching two ACUC pairs $ACUC_{Pref}$ taken from a provider's preferences and $ACUC_{Pol}$ specified in a consumer's policy essentially comes down to going over both trees simultaneously and verifying that it is possible to cover each branch of the policy-side tree with a more permissive preference-side branch. For instance, if $|ACUC_E| \trianglerighteq |ACUC_A|$ and $|ACUC_E| \trianglerighteq |ACUC_B|$ in Figures 6(b) and 6(a), then $ACUC_E \trianglerighteq ACUC_A$. However, it is impossible that $ACUC_A \trianglerighteq ACUC_E$ because $ACUC_E$ allows deeper downstream usage than $ACUC_A$.

Letting $UseDS(ACUC)$ denote a `UseDownStream` with an `ACUC` child element represented by $ACUC$, the matching for downstream usage rights works according to the



(a) Nested ACUC

(b) Recursive ACUC

(c) ACUC with multiple downstream

(d) Deeper ACUC

Figure 6: Examples of ACUC chaining

rule:

$$\begin{aligned} &UseDS(ACUC) \trianglerighteq UseDS(ACUC') \;\Leftrightarrow \\ &\quad (ACUC \trianglerighteq ACUC') \end{aligned} \tag{2.8}$$

2.3.3 Lazy matching

In the previous section we focused on *proactive matching*, i.e., when all downstream policies are known beforehand. It is however not always possible to collect all policies during matching. For this reason, we also introduce *lazy matching*, which only takes into account the properties and policies of the data consumer himself, but not those of any downstream data consumers. Rather, the data consumer expresses that he's willing to impose whatever usage restrictions on downstream consumers that the data provider may specify.

Both types of matching imply that the sticky policy that the data controller associates to the data must at least enforce the preferences of the data provider. On one hand, proactive matching makes it possible to minimize the authorizations and maximize the obligations that are transferred, since the matching procedure can already take the downstream consumers and their policies into account. On the other hand, lazy matching offers more flexibility and is the only option in dynamic settings, where either the downstream consumers or their policies are not known yet at the moment of matching, or where the access control policy depends on environment variables that will only be known when the data is actually forwarded.

To support lazy matching, we redefine formula 2.8 to take into account the `allowLazy` attribute, represented by a boolean value *lazy* here. Matching for downstream usage then follows the rule:

$$\begin{aligned} &UseDS(lazy, ACUC) \trianglerighteq UseDS(lazy', ACUC') \;\Leftrightarrow \\ &\quad (lazy \wedge lazy') \;\vee\; (ACUC \trianglerighteq ACUC') \end{aligned} \tag{2.9}$$

2.4 Future research

In the second year of the project we focused on the definition of a simple yet expressive language to specify privacy policies, user preferences, and to help users deciding whether personal data can be shared according to policies and preferences. A key advantage of this language is the bounded complexity of the matching algorithm that does not result in constraint satisfaction.

Since we mainly focus on the downstream aspect of privacy matching, only basic examples of rights and obligations are provided. More details can be found in [Rag09]. The work will be further extended as described below.

- *Specification of logic-based representation of matching.* Specifying the rules in a more formal way, e.g. using First Order Logic (FOL), Description Logic (DL) [BHS07], Formula [JSS08], or the Obligation Specification Language (OSL) [PSSW08] would be useful to verify matching, to reason on causes of mismatch, and to propose solutions (e.g., modified preferences). We started investigating different options and will continue this work in WP5.2.

- *Composition of policies.* When combining pieces of data we can either keep separate preferences (or sticky policies) or compose preferences. In this case, each individual preference must be more (or equally) permissive than the composite preferences. When specifying the policy of a front end service, it is possible to either specify the policy of each downstream service or to combine those downstream policies into one composite policy. In this case, the composite policy must be more permissive than each individual policy. Specifying policy composition will be done in WP5.2 and WP6.3.

- *Obligation and authorization ontology.* To be fully functional, one has to use our language in combination with a more complete ontology of authorization and obligation types. The definition of such an ontology is out of scope for this work, but we are planning to define more obligation types in a prototype implementation.

- *Integration into XACML.* We consciously kept our policy language rather simple in expressing the applicability of a rule. By embedding our language into a practical language such as XACML, we could leverage the higher expressivity of the latter for our purposes. One issue that remains to be solved in this case, however, is what effect XACML's rule-combining algorithms will have on the embedded policies, in particular for combining algorithms where the effects of multiple rules have to be taken into account simultaneously. For some of these, one may have to compose the policies of multiple rules, similarly to what was suggested above when combining multiple pieces of data.

- *Matching of access control policies.* By embedding our language into XACML, one can also profit from the expressivity of the latter in access control restrictions. One problem in that case, however, is that it becomes much harder to proactively match a downstream data consumer's properties against a specified access control policy, because not all the relevant attributes of the downstream consumer may be known upfront. In particular, environment attributes (e.g., time of day, server

load) may only be known at the time of actual access, further complicating the proactive matching procedure.

One idea to resolve this could be to design a "hybrid" between lazy and proactive matching, where both the data consumer's usage control policy does not contain the properties of the downstream consumer, but rather contains an access control policy that will be enforced on downstream consumers. Matching with a data provider's preferences involves checking whether the access control policy in the consumer's policy implies the one in the provider's preferences. Efficiently deciding implication of two access control policies may not be trivial for practical languages, however.

- *Comparing authorization and data sharing.* In this work we focus on scenarios where the data is shared with the data consumer that must enforce usage control. In on-line scenarios, this could be implemented by keeping the data at user side and requiring the data consumer to request the data, for a specific purpose, each time it requires the data and to delete it immediately after usage. This approach would provide a better user control since the user would enforce usage control and be able at anytime to update personal data or to revoke access. Downstream sharing would be instantiated as delegation of rights, i.e. the data consumer authorizes a third party to access the user's personal data for a specific purpose. We want to evaluate whether the proposed language can be reused in such scenarios.

Chapter 3

Legal policy mechanisms (Task 5.2.3)

Goals of the research in Task 5.2.3 are especially the description and analysis of legal requirements for privacy policies, thus building the basis for technical requirements to fulfill these legal parameters. The results shall lead to a new generation of technically supported privacy policies, that implement legal requirements in a suitable way. One focus of the research is the way of implementing the legal requirements by the usage of "sticky policies". Sticky policies define how data can be handled and "stick" with the data, when it travels or is processed. It allows for describing authorizations, that is, to define what the data controller can do with the collected data and under which conditions data can be shared with others. Also, it describes the obligations that the data controller has to obey on the data.

The current completed research has addressed the problem of analyzing existing policies languages and of developing models, languages, and policies supporting complex privacy requirements emerging in different contexts, where there is the need of integrating different sources of personal information. As a first step, use cases, privacy policies, and legal clauses have been analyzed and the results have been compared with the legal requirements in selected cases.

3.1 Legal requirements

General legal requirements for processing personal data are defined in the Data Protection Directive 95/46/EC (DPD) and the Directive on Privacy and Electronic Communications 2002/58/EC (DPEC). DPD and DPEC mandate member states of the EU to implement its normative content into their respective jurisdictions. The state of the implementation is documented by the EDPS (European Data Protection Supervisor).

Legal requirement for privacy policies encompasses data collection as well as data handling, i.e., the processing. Core element of the DPD is that collecting and processing of personal data is only allowed, if there is a legal basis or if the person concerned

unambiguously gave her consent, cf. Article 7 of the DPD. If the collection is based on the latter, the following information has to be provided to the data subject (amongst others):

- the data controller has to be declared as well as
- the types of data collected,
- and legitimate purpose of collecting data has to be defined, Article 6 and 7 of the DPD.

In the further process, the data controller has to ensure that the data is kept accurate and only used for the purposes declared ex ante. Thus, the amendment of the German Federal Data Protection act (BDSG) addicted a special part for the rules of order data processing. While the purposes of data processing are relatively easy to define, a specific problem of the legal requirements is defining of the purpose of data collecting. There are different positions regarding how the purpose has to defined.

- The first position acts on the assumption that the purpose has to be defined exactly. Therefore each privacy policy that does not define the purpose of collecting data would be insufficient. This can for example be derived from the legal text of the German Federal Data Protection act (BDSG), article 28, deposit 2, which implements the DPD in Germany.

How precise this has to be - according to this assumption - can be seen by the following: If the data controller wants to use the data for a different purpose than she did before collecting data and if she displayed this in his policy before, a new consent of data subject is needed. Otherwise use would only be allowed if the data collector has a qualified interest in using data for a different purpose (and if these interests are of a higher value than the interests of the person concerned). This would lead to the interpretation that the purpose of data collecting has to be defined quite precisely (see [GS07], BDSG, article 28, marginal number 48), how precise exactly then needs further elaboration. Thus, this position acts on the maxime that each step must be displayed exactly.

- The second position acts on the assumption that the requirement of Article 7 of the DPD, which demands the necessarily of collecting data for the purpose of a contract does not touch the freedom of contract. Therefore a privacy policy only has to define which kind of data is collected for fulfilling a contract between the data controller and the person concerned. The main argument is that data collection for the performance of a contract is purpose enough (see [SS08], BDSG, article 28, marginal number 4). The inherent problem with this is that only the purpose would be declared, but that there would be a lack of defining the applicability of the data collection.
- A third, coupling, position combines the approaches of position 1 and position 2 and therefore acts on the assumption that the maxim of freedom of contract can be kept even if the purpose of collecting data is defined in an exact way. That means that the purpose of collecting data should be defined as precise as necessary

but that there is no need of defining exactly each step of collecting data. Only the purpose has to be displayed exactly. Therefore purpose and applicability of data collection is transmitted and displayed, if it is necessary and applicable for the performance of a contract (see [GS07], BDSG, article 28, marginal number 13).

The analyzed privacy policies, as well as legal sources for collection and processing, show differing results. Our research indicates that to improve the state of art of displaying data collecting purposes and to show which way of displaying the purpose of collecting data fits best to legal requirements further research is needed. The analyzed policies are part of a not yet published internal deliverable, thus they are not a part of this document.

3.2 Transferring legal into technical requirements

In a second step, research on the above mentioned issues is applied towards the technical framework in PrimeLife. Highly relevant is the research on Policy Languages, but also the work on Human Computer Interaction (HCI). The former work largely relies on Access Control Mechanisms (or Policies, ACP) and Data Handling Mechanisms (or Policies, DHP). According to the fact that transparency is one of the core principles of data protection legislation in Europe (DPD) beyond and all around the world (see [OEC80]), the person concerned should be aware of "who knows what about her". Therefore in using ACP, the first or the third position towards displaying the purpose of data collecting in privacy policies seem to be preferable.

To improve the state of art, further research on different kinds of policies should be performed in use-cases like privacy policies or legal text. However the results of the completed research according this topic vary and do not yet give clear indication on how to best support the technical mechanisms. Some of the policies implemented the legal requirements well, others did it different (maybe, because they had different understanding of the legal requirements and acted on the assumption, that they fulfilled the legal requirements).

This accumulates to two questions:

1. with relevance for HCI, the displaying of data handling in a privacy policy would lead to the fact that the privacy policy is getting quite long. If a look is taken to the fact that the amount of time required to read privacy policies is too great anyhow (see [GPSC09]). Displaying also the internal data handling in the policy would make it even longer.

2. according to the positions 1 and 3 towards ACP, the internal data handling has to be described as well. But the displaying of internal data handling in a privacy policy may touch questions of business secrets. On the other hand, the internal data handling has to fulfill legal requirements as well. This could be implemented by the usage of sticky policies for this part of the privacy policy. The sticky policy is an agreement between the data subject and the data controller on the handling of personal data collected from the data subject (see [Rag09]).

3.3 Applicability to the state of the art in policy languages

The difficulty of determining how precise the policy language has to be can also be seen in light of the DPD. According to article 12, lit a, bullet point 3, the data subject has the right of information about the data handling. This also is the expression of the right of informational self-determination of the data subject. Therefore it is necessary that the data collector displays at least applied information about data handling in her policy. And the data collector also needs a precise policy to comply with its promises when internally processing the data in its own business. To ensure the implementation of legal requirements during data handling, the data collector has to advice her associates on how to process with data collected. To avoid mistakes in handling data, the advice has to be as precise as possible.

A privacy policy that displays this, implements legal requirements for the associates of the data collector as well as for the data subject. The maxim of transparency in data collecting and data handling also argues to the assumption that a privacy policy has to be as precise as possible. This clarifies the need of precise and well implemented privacy policies.

In the following, we put related technological approaches in the context of legal research.

XACML and XACML-PrimeLife. The extensible access control markup language (XACML) is an XML-based language for expressing and interchanging access control policies, which is now being extended by PrimeLife for adding new privacy-related features such as anonymous credentials and data handling (see Section 1.3). Such extended XACML can be used by the data controller as well as by the person concerned. The result of the completed research towards XACML-PrimeLife is the fact that on one hand it is a suitable way to display the data collecting party, but, on the other hand, there is still difficulty in displaying the purpose of the data collecting. Currently, the research forms a wrapper in this regard, that can be used to contain elements of the Platform for Privacy Preference (P3P), but also could include an ontology.

P3P. P3P enables Websites to express their privacy practices in a standard format that can be retrieved automatically and interpreted easily by user agents (see `http://www.w3.org/TR/P3P/`). P3P user agents will allow users to be informed of site practices (in both machine-and human-readable formats) and to automate decision-making based on these practices when appropriate (see Section 2.1.2). Therefore users do need to read the privacy policies at every site they visit. Problematic however is the transparency in using P3P, it seems to be to complicated to be understandable for users. But P3P also has the problem of describing the aspects most relevant to the user: what is actually going to happen to the data, what purposes are they used for and under which conditions and with what obligations? (see [Sam09]). Moreover the expressive power of P3P especially in regard to describing purposes is limited (albeit extensible). The predefined set of purposes was limited to so-called secondary purposes in the first specification (`http://www.w3.org/TR/P3P/`), which was then complemented with a flat partonomy of some twenty "primary purposes" in version 1.1 (`http://www.w3.org/TR/P3P11/#ppurpose`. Taking into account the requirements for displaying the purpose to the user, as well as

requirements of internally achieving legal processing of the data within the limits of the purpose, this appears to be too limited, and will need further extension, potentially by way of a more comprehensive ontology.

Liberty Alliance Identity governance framework. Another approach of solving this problem was Liberty's Identity Governance Framework (`http://www.projectliberty.org`). Privacy constraints describe fundamental constraints on the propagation, usage, retention, storage and display of identity data. There is concern regarding the appropriate use of identity data and privacy constraints permit the expression of constraints over the processing of data. Using policy frameworks, authorities and consumers can use privacy constraints to describe composite constraints on identity data. The liberty framework also has to deal with the problem of how precise the purpose of data collecting and data handling has to be displayed and how it is possible to display the policy for DHP as well as for ACP. There is also no ontology possible while using liberty framework.

3.4 Conclusion and interim results

Sticky policies promises to improve the state of the art in data protection, both on the level of better control, but also on the level of increasing transparency. The sticky policy is usually the result of an automated matching procedure between the data subject's data handling preferences and the data controller's data handling policy (see [ABD+09]). It is the agreed-upon sets of granted authorizations and promised obligations associated with a resource. Sticky policies are policies that control how data is to be assessed and used and that accompany data throughout an entire distributed system (see `http://www.cs.kuleuven.be/conference/MidSec2008/sticky.pdf`).

To clarify, how legal requirements for DHP and ACP can be displayed best, further research has to be done. Our research of a number of policies have shown different results on the usability of the tested policies. Some of them implemented the legal requirements for collecting data well, others did not. The implementation of legal requirements for data handling on the other hand were not displayed that well. The main problem in privacy policies seems to be the displaying of data handling. This also is legal requirement, but only few policies implemented it. So further research is needed, which could be done either in use-cases or in empirical studies.

It is due to the very nature of data processing that it is impossible to conduct a comprehensive ontology towards this topic, but there is the option of advanced taxonomy and possibly an ontology, which would cover processing to a certain level of detail and brevity.

3.5 Future research

There are still some questions in displaying legal requirements in policy languages unacknowledged. Further research will work on answering these questions and on finding solutions of best practicable displaying legal requirements in privacy policies to improve the state of art of this topic. To improve the state of art, the state of art still has to

be assessed. In the completed research, this was tried by different scientific approaches (see [Sam09]).

The research leads to the conclusion that actually many policies according data handling act on the assumption that it is enough to display the legitimate reason of the data collector on handling data as legal basis (see BDSG, article 28, deposit 1, number. 2 or article 7, deposit d of the DPD). This leads to the conclusion that the catchall element - legitimate reason - is used as general reason (see [SS08], BDSG, article 28, marginal number 1). This should not be the state of art of privacy policies. To avoid this for the future, a more precise description of policies is necessary.

The completed work includes different approaches of research, the research with use-cases as well as the research with empirical studies. Both approaches had the common goal in mind: the need of having access control policy languages that, on one hand, provide access control functionality and, on the other hand, protect the privacy of the involved parties and of their personal information.

It has to be stated that there has already been relevant effort with empirical approaches, which lead to some results, but yield the question of whether an extensive analysis, albeit promising, is achievable with reasonable efforts. To be more precise: Even the empiric analysis of only German data protection law and privacy policies of German sites would be immense. An analysis of published privacy policies yields similar limitations, especially when trying to effectively rule out bias of interpretation, the effort doubles or triples. The completed empiric research also leads to the conclusion that all analyzed approaches have deficits, like it is shown above. Developing the future research only on the analyzed approaches therefore would not lead to the best result.

Therefore empirical research will be supplemented by use-cased based analysis, where selected use cases are looked at from a legal perspective (i.e., what data is needed for what purposes, and in what processes). This approach does not promise the comprehensiveness of an empirical analysis, but rather focuses on an exemplification of the needed expressive power of the language. It includes, however, the advantage, of being able to look deeper into the technical processes underlying the respective policies return than solely looking at what is published in privacy policies. One focal point of future research therefore will be seen in research with use-cases. One of the benefits of this approach is the fact that it is easier to handle a known systems as a basis of research than to handle with unknown systems. Research with empiric studies would not lead to more exact results, because the use-case scenario can display all possible ways of collecting and processing data as well as all possible purposes and the necessary legal basis for it. One scenario that will display all this is the scenario of a web-shop, which is analyzed in the H5.2.2 deliverable. Nevertheless, the future research will take all possible options of approach.

Chapter 4

Abstracts of research papers

1. M. Ali, L. Bussard, U. Pinsdorf, "Obligation language and framework to enable privacy-aware SOA," in *4th International Workshop on Data Privacy Management (DPM 2009)* [ABP09].
   Abstract. Privacy policies defines rights and obligations on data (e.g. personally identifiable information) collected by services. Tackling privacy policies in a service oriented architecture spanning multiple trust domains is difficult because it requires a common specification and distributed enforcement. This paper focuses on the specication and enforcement of obligations. We describe the requirements, the resulting language, and its implementation. Finally, we compare our results with obligation support in the state of the art. The key contribution of this work is to bridge the gap between specific mechanisms to enforce obligations and underspecified support for obligations in today's access control and data handling policy languages.

2. C.A. Ardagna, J. Camenisch, M. Kohlweiss, R. Leenes, G. Neven, B. Priem, P. Samarati, D. Sommer, and M. Verdicchio, "Exploiting cryptography for privacy-enhanced access control: a result of the PRIME project," in *Journal of Computer Security (JCS)* [ACK+10].
   Abstract. We conduct more and more of our daily interactions over electronic media. The EC-funded project PRIME (Privacy and Identity Management for Europe) envisions that individuals will be able to interact in this information society in a secure and safe way while retaining control of their privacy. The project had set out to prove that existing privacy-enhancing technologies allow for the construction of a usercontrolled identity management system that comes surprisingly close to this vision. This paper describes two key elements of the PRIME identity management systems: anonymous credentials and policy languages that fully exploit the advanced functionality offered by anonymous credentials. These two key elements enable the users to carry out transactions, e.g., over the Internet, revealing only the strictly necessary personal information. Apart from presenting for the first time these two key results, this paper also motivates the need for privacy

enhancing identity management, gives concrete requirements for such a system and then describes the key principles of the PRIME identity management solution.

3. C.A. Ardagna, S. De Capitani di Vimercati, S. Paraboschi, E. Pedrini, and P. Samarati, "An XACML-based privacy-centered access control system," in *1st ACM Workshop on Information Security Governance (WISG 2009)* [ADP+09a].
Abstract. The widespread diffusion of the Internet as the platform for accessing distributed services makes available a huge amount of personal data, and a corresponding concern and demand from users, as well as legislation, for solutions providing users with form of control on their data. Responding to this requirement raises the emerging need of solutions supporting proper information security governance, allowing enterprises managing user information to enforce restrictions on information acquisition as well as its processing and secondary use. While the research community has acknowledged this emerging scenario, and research efforts are being devoted to it, current technologies provide still limited solutions to the problem. In this paper, we illustrate our effort in pursuing the goal of bringing information security governance restrictions deployable in current organizational contexts. Considering the large success and application of XACML, we extend the XACML architecture and modules complementing them with functionalities for effective credential-based management and privacy support. Our proposal combines XACML with PRIME, a novel solution supporting privacy-aware access control, resulting in an infrastructure that provides the flexible access functionality of XACML enriched with the data governance and privacy features of PRIME.

4. C.A. Ardagna, S. De Capitani di Vimercati, E. Pedrini, S. Paraboschi, P. Samarati, and M. Verdicchio, "Extending XACML for open web-based scenarios," in *W3C Workshop on Access Control Application Scenarios* [ADP+09b].
Abstract. Traditional access control solutions, based on preliminary identification and authentication of the access requester, are not adequate for open Web service systems, where servers generally do not have prior knowledge of the requesters. In this paper, we provide some extensions to the eXtensible Access ControlMarkup Language (XACML), which is the most significant and emerging solution for controlling access in an interoperable and flexible way, to make it easily deployable and suitable for open Web-based systems.

5. C.A. Ardagna, L. Bussard, S. De Capitani di Vimercati, G. Neven, E. Pedrini, S. Paraboschi, F. Preiss, P. Samarati, S. Trabelsi, and M. Verdicchio, "PrimeLife policy language," in *W3C Workshop on Access Control Application Scenarios* [ABD+09].
Abstract. The sudden popularity of social networks and web 2.0 applications changed radically the Internet landscape and the users' behavior. Today's young people are the first generation with the ability to distribute information quickly, cheaply and to large groups of people. The amount of personal and private information published and stored in the servers becomes so huge that the traditional concepts of privacy were radically affected. To appease such concerns, enterprises and service providers publish privacy statements that promise fair information practices. Written in natural language or formalized using languages like P3P, EPAL,

XACML etc... they are only promises but not necessarily enforced by technical measures. These problems are amplified if personal data is used not only by the enterprise that collected the data, but also by secondary users such as partner organizations, or government agencies. These flows of data are complex. Threats to data privacy can come from inside (accidental disclosure, insider curiosity and subornation) as well as from the outside (uncontrolled secondary usage) of each organization. Putting customer information online further increases the risk of exposing private and sensitive information to outsiders. In this paper we propose a new policy language handling access control and data usage at the same time. In the context of the European ICT PrimeLife we propose an extension of the eXtensible access control markup language (XACML 3.0) offering one of the most popular standardized policy language. This extension suggests a new obligation handling mechanism taking into account temporal constraints, pre-obligations, conditional obligations, and repeating obligations together with a down-stream usage authorization system defining the access control rules under which personal information collected by an entity can be forwarded to a third party. Moreover, our language is based on the concept of trusted credentials.

6. P. Bichsel, S. Mueller, F. Preiss, D. Sommer, M. Verdicchio, "Security and trust through electronic social network-based interactions," in *Workshop on Security and Privacy in Online Social Networking* [BMP+09].
   Abstract. The success of a Public Key Infrastructure such as the Web of Trust (WoT) heavily depends on its ability to ensure that public keys are used by their legitimate owners, thereby avoiding malicious impersonations. To guarantee this property, the WoT requires users to physically gather, check each other's credentials (e.g., ID cards), to sign the trusted keys, and to subsequently monitor their validity over time. This trust establishment and management procedure is rather cumbersome and, as we believe, the main reason for the limited adoption of the WoT. To overcome this problem, we propose a solution that leverages the intrinsic properties of Electronic Social Networks (ESN) to establish and manage trust in the WoT. In particular, we exploit dynamically changing profile and contact information, as well as interactions among users of ESNs to gain and maintain trust in the legitimacy of key ownerships without the disadvantages of the traditional WoT approach. We see our proposal as an effective way to make security and trust solutions available to a broad audience of non-technical users.

7. J. Camenisch, S. Mödersheim, G. Neven, F.-S. Preiss, and D. Sommer, "Credential-based access control extensions to XACML," in *W3C Workshop on Access Control Application Scenarios* [CMN+09].
   Abstract. Access control and authentication systems are currently undergoing a paradigm shift towards openness and user-centricity where service providers communicate to the users what information they need to provide to gain access to a given resource. This paradigm shift is a crucial step towards allowing users to manage their identities and privacy. To ensure the service provider of the validity of the presented information, the latter is typically attested to by a trusted issuer or identity provider. There are multiple means to transmit such attestation to the service provider including X.509 certificates, anonymous credentials, and

OpenIDs. In this position paper, we advocate to abstract all attestation means into the concept of a 'credential' and propose to extend XACML so that it allows service providers to specify the set of credentials that a user is required to present to get access to a given resource. Our extensions not only allow one to express conditions on the credentials that the user has to present, but also which attributes have to be disclosed, and to whom, and which statements the user has to consent to before being granted access.

Bibliography

[ABD+09] C.A. Ardagna, L. Bussard, S. De Capitani di Vimercati, G. Neven, E. Pedrini, S. Paraboschi, F. Preiss, P. Samarati, S. Trabelsi, and M. Verdicchio. Primelife policy language. In *W3C Workshop on Access Control Application Scenarios*, Luxembourg, November 2009.

[ABP09] M. Ali, L. Bussard, and U. Pinsdorf. Obligation language and framework to enable privacy-aware SOA. In *Proc. of the 4th International Workshop on Data Privacy Management (DPM 2009)*, Saint Malo, France, September 2009.

[ACDS08] C.A. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati. A privacy-aware access control system. *Journal of Computer Security (JCS)*, 16(4):369–392, 2008.

[ACK+10] C.A. Ardagna, J. Camenisch, M. Kohlweiss, R. Leenes, G. Neven, B. Priem, P. Samarati, D. Sommer, and M. Verdicchio. Exploiting cryptography for privacy-enhanced access control: A result of the prime project. *Journal of Computer Security (JCS)*, 18(1):123–160, 2010.

[ADP+09a] C. A. Ardagna, S. De Capitani di Vimercati, S. Paraboschi, E. Pedrini, and P. Samarati. An XACML-based privacy-centered access control system. In *Proc. of the 1st ACM Workshop on Information Security Governance (WISG 2009)*, Chicago, Illinois, USA, November 2009.

[ADP+09b] C.A. Ardagna, S. De Capitani di Vimercati, E. Pedrini, S. Paraboschi, P. Samarati, and M. Verdicchio. Extending xacml for open web-based scenarios. In *W3C Workshop on Access Control Application Scenarios*, Luxembourg, November 2009.

[AHK+03] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise privacy authorization language (EPAL 1.2), 2003. `http://www.w3.org/Submission/EPAL/`.

[AHKS02] 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.

[AL04] A. Anderson and H. Lockhart. *SAML 2.0 profile of XACML*. OASIS, September 2004.

[BCC04] E.F. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *Proc. of the 11th ACM Conference on Computer and Communications Security (CCS 2004)*, Washington, DC, USA, October 2004.

[BHS07] F. Baader, I. Horrocks, and U. Sattler. Description logics. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*. Elsevier, 2007.

[BMB09] M.Y. Becker, A. Malkis, and L. Bussard. A framework for privacy preferences and data-handling policies. Technical Report MSR-TR-2009-128, Microsoft Research, September 2009.

[BMP+09] P. Bichsel, S. Mueller, F. Preiss, D. Sommer, and M. Verdicchio. Security and trust through electronic social network-based interactions. In *Proc. of the Workshop on Security and Privacy in Online Social Networking*, Vancouver, Canada, August 2009.

[Boa07] S. Boag et al. *XQuery 1.0: An XML Query Language*. World Wide Web Consortium (W3C), 2007.

[Bra99] S. Brands. *Rethinking Public Key Infrastructure and Digital Certificates — Building in Privacy*. PhD thesis, Technical University Eindhoven, 1999.

[BS02] P. 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.

[Cha85] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[CHK+06] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *Proc. of the 13th ACM Conference on Computer and Communications Security (CCS 2006)*, Alexandria, VA, USA, October-November 2006.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proc. of EUROCRYPT 2001*, Innsbruck, Austria, May 2001.

[CMN+09] J. Camenisch, S. Mödersheim, G. Neven, F.-S. Preiss, and D. Sommer. Credential-based access control extensions to xacml. In *W3C Workshop on Access Control Application Scenarios*, Luxembourg, November 2009.

[Con02] ContentGuard. XrML 2.0 Technical Overview. `http://www.xrml.org/reference/XrMLTechnicalOverviewV1.pdf`, 2002.

[CS03] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Proc. of the 23rd Annual International Cryptology Conference*, Santa Barbara, CA, USA, August 2003.

[DFJS07] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, and P. Samarati. Access control policies and languages in open environments. In T. Yu and S. Jajodia, editors, *Secure Data Management in Decentralized Systems*. Springer-Verlag, 2007.

[eXt05] *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`.

[GPSC09] J. Gomez, T. Pinnick, A. Soltani, and B. Carver. Knowprivacy, June 2009. `http://www.knowprivacy.org/report/KnowPrivacy_Final_Report.pdf`.

[GS07] P. Gola and R. Schomerus. Bundesdatenschutzgesetz - Kommentar (accompanying commentary), 2007.

[HBP05] M. Hilty, D. Basin, and A. Pretschner. On obligations. *Lecture Notes in Computer Science*, 3679:98–117, 2005.

[JSS08] E.K. Jackson, W. Schulte, and J. Sztipanovits. The power of rich syntax for model-based development. Technical report, Microsoft Research, 2008.

[Leu07] M. Leumann. Policy evaluation and negotiation in distributed usage control (Master Thesis), 2007.

[ODR02] ODRL. Open Digital Rights Language (ODRL), version 1.1, 2002.

[OEC80] OECD. *OECD guidelines on the protection of privacy and transborder flows of personal data*. Organisation for economic Co-operation and Development, 1980.

[PS04] J. Park and R. Sandhu. The UCONABC usage control model. *ACM Transactions on Information System Security*, 7(1):128–174, 2004.

[PSSW08] A. Pretschner, F. Schütz, C. Schaefer, and T. Walter. Policy evolution in distributed usage control. In *Proc. of the 4th International Workshop on Security and Trust Management (STM 2008)*, Trondhem, Norway, June 2008.

[Rag09] D. Raggett. Draft 2nd design for policy languages and protocols. Technical Report H5.3.2, PrimeLife project, July 2009.

[Sam09] P. Samarati. First research report on research on next generation policies. Technical Report D5.2.1, PrimeLife project, February 2009.

[SD01] 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.

[SK07] N. Seki and M. Kudo. Access control policy for XML. In M. Gertz and S. Jajodia, editors, *Handbook of Database Security: Applications and Trends*. Springer-Verlag, 2007.

[SS08] G. Spindler and F. Schuster. *Recht der Elektronischen Medien.* Verlag C. H. Beck München, 2008.

[W3C02] W3C. *A P3P Preference Exchange Language 1.0 (APPEL1.0)*, April 2002. `http://www.w3.org/TR/P3P-preferences/`.

[W3C06] W3C. The Platform for Privacy Preferences 1.1 (P3P1.1) Specification, 2006. `http://www.w3.org/TR/P3P11/`.

[Wan04] X. Wang. MPEG-21 rights expression language: Enabling interoperable digital rights management. *IEEE Multimedia*, 11(4):84–87, 2004.