Privacy and Identity Management in Europe for Life

Final research report on next generation policies

Editors:	Sabrina De Capitani di Vimercati (UNIMI)
	Pierangela Samarati (UNIMI)
Reviewers:	Michele Bezzi (SAP)
	Fatih Gey (EMIC)

Identifier:	D5.2.3
Type:	Deliverable
Version:	1.0
Class:	Public
Date:	May 19, 2011

Abstract

The open and dynamic nature of today's Information Society requires the development of novel privacy-aware paradigms for access control policies and languages. Goal of Work Package 5.2 is the definition of flexible and comprehensive privacy-aware policies actually deployable with today's technology.

This document presents the advancements made in the research work during the third year of PrimeLife in Work Package 5.2. The document includes one chapter for each task of the work package, briefly describing the main research results, along with a brief description of plans for continuing the work after the end of the project. The last chapter lists the abstracts of the research papers, reporting the findings of Work Package 5.2, published in the third 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,IBM,ULD,UNIBG,UNIMI

Executive Summary

The widespread access to information enabled by the Information and Communication Technologies (ICTs) brings significant benefits allowing users to access electronic services and resources anywhere anytime. These advantages come at a price of higher privacy risk as a huge amount of (private) information is being shared and stored, often without direct control of its owner. The definition of access control models and languages for giving users effective control on their private information when interacting with other parties, deciding what to release and imposing possible conditions on how such information should be handled by the recipient, is therefore one of the most challenging problems of today's society. This problem has been under the attention of the research and development communities and several investigations have been carried out, proposing novel privacy-aware policies, models, and languages for emerging scenarios. In particular, work has aimed at departing from user authentication and providing access control solutions supporting credential-based and attribute-based specifications. Although these studies have brought important advancements in the privacy-aware policy and language area, they tackle only part of the problem. Different aspects still need to be investigated, including: effective support for the user in specifying (and having enforced) possible privacy preferences on information to be released when interacting with other parties; solutions for servers to manage privacy policies, detect and solve possible mismatches and conflicts between them; and a good understanding of legal requirements for enabling technical solutions complying with them.

With the observations above in mind, goal of Work Package 5.2 is the definition of a flexible and comprehensive approach to privacy policies. In particular, Work Package 5.2 aims at providing the basis for an effective and easily deployable privacy-aware policy solution that can be adopted in current information systems with minimal impact on existing technologies. The Work Package is organized in three tasks: *Task 5.2.1 – Policy languages* focuses on the definition of policy languages able to protect the user's privacy; *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.

For each task of the Work Package 5.2, this document presents the research work carried out in the third year of the project, which can be summarized as follows.

- *Task 5.2.1* addressed the problem of developing a powerful, flexible, and easy to use solution for specifying privacy preferences to offer clients a more meaningful control over information released to a server. The task also provided a formal modeling of the concepts that have to be supported for enforcing the new access control paradigm needed in open scenarios, and proposed a flexible solution allowing servers to specify how their policies should be communicated to clients.

- *Task 5.2.2* focused on privacy mismatches, obligations, and privacy policies in Service-Oriented Architectures (SOAs). The task analyzed the privacy mismatches that may arise when multiple layers of abstraction exist to represent privacy preferences and privacy policies. The task also studied the dependencies between authorizations and obligations with the goal of avoiding ambiguous interactions between them. Finally, the task defined an abstract framework to reason on privacy policies in SOAs.

- *Task 5.2.3* developed new ways to strengthen the protection of personal data by better understanding the underlying legal requirements. The task developed a legal framework for processing personal data, existing policy languages, and their potentials and shortcomings, as well as the different approaches of finding a suitable policy language that matches the different use cases. The results will lead to a new generation of technically supported privacy policies that implement legal requirements in a suitable way.

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 third year of the project within Tasks 5.2.1, 5.2.2, and 5.2.3, respectively. The last chapter (Chapter 4) lists the abstracts of the research papers reporting the findings of Work Package 5.2 published in the third year of the project.

Contents

List of Figures

List of Tables

Chapter 1

Policy languages (Task 5.2.1)

Despite the great improvements of ICT systems in access to information and communication, there are important user-side and server-side requirements that are currently not completely satisfied in policy language definition and implementation. On one hand, users want to regulate disclosure of their credentials and properties when interacting with servers in open scenarios. On the other hand, service providers need a simple, expressive, and flexible language to specify access control policies that suit the requirements introduced by open scenarios. The Chapter illustrates the research results of Task 5.2.1 aimed at supporting privacy-aware client-server interactions, and for user empowerment.

1.1 Introduction

Advancements in the Information Technology allow unknown parties to interact over the Internet, to the aim of offering/accessing resources. In such *open scenarios*, selective access to resources is typically based on the evaluation of (certified or uncertified) properties that clients requesting access present to the servers offering resources [BS02]. Since the client and the server may be unknown to each other, the client cannot know which properties she should present to acquire access. Analogously, the server does not know the client properties to decide if access should be permitted or denied. It is therefore necessary for the server to disclose its access control policies and for the client to release a set of credentials certifying her properties. Since both client credentials and server access control policies may contain sensitive information, their release must minimize the disclosure of unnecessary information. The development of a privacy-preserving system for open scenarios requires to address several issues at both the server and client side, also empowering users to specify restrictions on information they possibly release to servers to the purpose of obtaining access. The remaining of this chapter is organized as follows. Section 1.2 presents client-side solutions for regulating the release of private personal information (i.e., credentials). The definition of an expressive and flexible approach for regulating the release of client personal data requires a fine-grained modeling of the client portfolio and the definition of simple tech-

niques that permit the client to easily manage each portfolio element according with her disclosure preferences [ADF+11, ADF+10c, ADF+10d, ADF+10e]. Section 1.3 first describes novel concepts that have to be supported by an access control system suitable for open scenarios, to the aim of providing an expressive solution actually deployable with today's technology. It then describes a server-side solution that permits the server to selectively disclose its policies, which may contain sensitive data that must be kept confidential, while guaranteeing the client to have enough information for possibly gaining access [ADF+10b, ADP+11, ADN+10].

1.2 Supporting privacy preferences in credential-based access control

Access control solutions for open systems are typically based on the assumption that a client may adopt approaches specifically designed for the server to protect the disclosure of her sensitive information. These solutions however do not consider the specific privacy requirements characterizing the client. This section describes a novel approach that puts forward the idea of adopting a different model at the client-side, aimed at minimizing the amount of sensitive information released to a server. This model is based on a formal modeling of the client portfolio and easily supports the definition of privacy preferences and disclosure limitations for empowering the user in the release of her personal information.

1.2.1 Modeling of the client portfolio

A key requirement for a framework empowering a client with full control over personal information released during the interactions with a server is the definition of an expressive model representing such a personal information. The model should enable the client to easily state her preferences on the disclosure of credentials/properties and should take advantage from emerging credential technologies that support the selective release of properties within credentials, as allowed by novel anonymous credentials (e.g., [AL04, HBH07]). In the following, the term *portfolio* will be used to refer to the set of certified properties (*credentials*) and uncertified properties (*declarations*) characterizing a client [BS02].

- *Abstractions over credential types.* Each credential in the client portfolio is characterized by a unique identifier, a set of certified properties, an issuer, and a type. The type of a credential identifies the properties that the credential certifies. The declarations form a special credential of type "declaration" whose issuer is the client itself. Abstractions can be defined over the credential types, possibly introducing a hierarchy of types. Formally, a hierarchy $\mathcal{H}$ of credential types is a pair $(\mathcal{T}, \preceq_{isa})$, where $\mathcal{T}$ is the set of all credential types and abstractions over them, and $\preceq_{isa}$ is a partial order relationship over $\mathcal{T}$. Given two types t_i and t_j in $\mathcal{T}$, $t_i \preceq_{isa} t_j$ if t_j is an abstraction of t_i. Hierarchy $\mathcal{H}$ has a unique root, denoted with $*$, such that $t_i \preceq_{isa} *$, for each $t_i \in \mathcal{T}$. Figure 1 illustrates a hierarchy of credential types where, for example, *id* is an abstraction of credential types *id_card* and *dr_license* (i.e., *id_card* $\preceq_{isa}$ *id* and *dr_license* $\preceq_{isa}$ *id*). The hierarchy of credential types represents

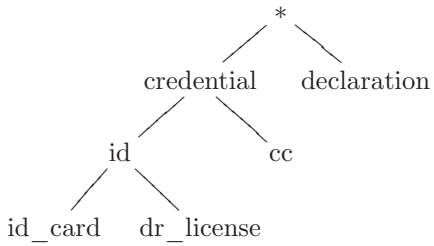


Figure 1: An example of hierarchy of credential types

the common knowledge shared between a client and a server. Since a server has not any knowledge about the client portfolio, any request by the server for a set of (certified) properties may only refer to credential types and abstractions in the hierarchy.

- *Atomic vs non-atomic credentials.* Atomic credentials can only be released as a whole, that is, their release entails the disclosure of all the properties they certify. By contrast, non-atomic credentials support the selective release of individual properties extracted from the credential. Atomic credentials (e.g., X.509 certificates) are the most common kind of credential used today in distributed systems. By contrast, non-atomic credentials (e.g., U-Prove and Idemix [Bra00, CL01]) are based on modern technologies and permit also to certify the possession of an instance of a given credential type, without disclosing the properties within it. Clearly, declarations are non-atomic credentials.

- *Credential-independent vs credential-dependent properties.* Properties in the client portfolio can be associated uniquely with the client or can depend on the specific credential certifying them. For instance, name is a property that can be certified by different credentials but its value depends only on the owner of these credentials. A property representing a credit card number can instead be specific of the credential certifying it. In the following, the terms *credential-independent* and *credential-dependent* will be used to refer to properties whose values depend only on the credential owner and on the specific credential certifying them, respectively.

The client portfolio can be conveniently modeled as a *portfolio graph*, which is a bipartite graph having a vertex for each credential and each property in the portfolio, and an edge connecting each credential to the properties contained in it. Figure 2 illustrates an example of a portfolio graph, where credential are represented as rectangles and property are represented as ovals. The portfolio includes four credentials, *myId* (of type id_card), *myLicense* (of type dr_license), *myVISA* and *myMC* (both of type cc), and a declaration *decl*. The only non atomic credential is *myLicense*, while all the other credentials are atomic. In the figure, atomic credentials can be distinguished from non atomic credentials since all the edges incident to an atomic credential are attached to a black semicircle.

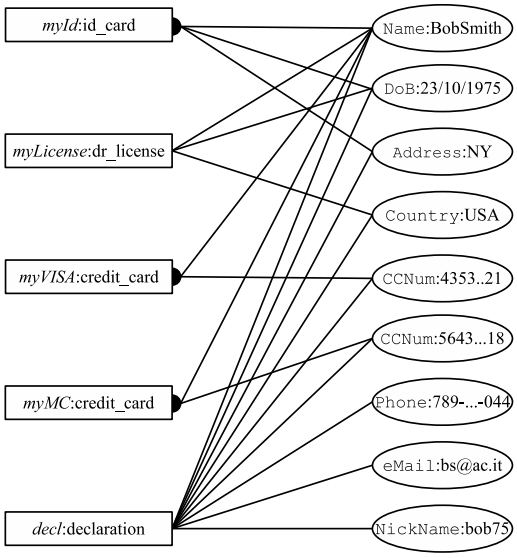


Figure 2: An example of a portfolio graph

1.2.2 Portfolio sensitivity

When accessing a service/resource, a client may need to release personal information according to a server request, stating the information about which conditions need to be satisfied for the access to be granted. A client has then to determine which properties/-credentials disclose to gain access while also limiting the release of sensitive information that is not strictly necessary. For instance, suppose that for accessing a specific service, a server requires the address or the phone number of the requesting client. In this case, the client may prefer to release her address instead of her phone number since she considers the phone number more sensitive than the address. Clearly, if there are several alternative options from which the client can choose, the task of determining which properties/credentials release may become complex. A simple, while effective and flexible, solution to express privacy preferences on the release of properties/credentials is then needed to: *i)* automatically regulate the disclosure of sensitive information upon server requests; and *ii)* preserve the privacy of the client by providing a support for determining the minimal information that has to be released to acquire a service.

Information privacy preferences on a portfolio can be specified in terms of *sensitivity labels* that the user can associate with the different components (or combinations thereof) in her portfolio. The set Λ of sensitivity labels could be any set of values, provided the existence of a (partial) order relationship $\succeq$ over them and a composition operator $\oplus$ that determines the label resulting from the combination of two labels. This generic definition of sensitivity labels permits to capture different ways of expressing preferences. For instance, sensitivity labels could be classical multilevel security classifications (e.g., Top Secret, Secret, Confidential, Unclassified, possibly with associated categories) with the

$\oplus$ operator corresponding to the *least upper bound*. Also, they could be positive integer values, where the $\oplus$ operator can be either the *sum* (i.e., $\lambda_i \oplus \lambda_j = \lambda_i + \lambda_j$) and therefore reflect an *additive property*, or the *maximum* (i.e., $\lambda_i \oplus \lambda_j = max(\lambda_i, \lambda_j)$). In the following, for simplicity, Λ corresponds to the set $\mathbb{Z}$ of positive and negative integer values. Note that a user may specify different preferences for different servers, by assigning a different label to each component in its portfolio, depending on the server requesting the release. The request is then evaluated on the instance of the labels determined by the server with whom the user is interacting [ADF+11].

Sensitivity of properties and credentials. The first step for the user to specify how much she values information in her portfolio is to associate a sensitivity label λ with each property p and credential c.

- $\lambda(p)$: reflects how much the user considers property p sensitive and therefore how much she values its release.
- $\lambda(c)$: defines the sensitivity of the *existence* of a credential. This is the additional information carried by the credential itself, regardless of the information contained in it. For instance, a dialysis certificate may include only properties `Name` and `Address`, but the existence of the certificate itself has an additional sensitivity that goes beyond the demographic information of the user.

Sensitivity of associations. In general, releasing a set of elements entails a sensitivity corresponding to the combination of the sensitivity of all the elements involved. There are however cases where merging some elements might produce an information release that does not precisely correspond to the composition of the labels of the individual elements. Let a be any set of properties and/or credentials. The user can specify $\lambda(a)$ as the *additional*, positive or negative, sensitivity to take into account in computing the sensitivity of the set of elements jointly released.

- *Sensitive views* (*positive* $\lambda(a)$). They reflect the fact that a set of portfolio elements jointly released carries *more* information than the composition (i.e., the sum) of the labels of the individual elements. For instance, the association between a user `Address` and one of her credit cards (e.g., *MyVISA*) can be considered more sensitive than the composition of the sensitivity labels of property `Address` and credential *MyVISA*.
- *Dependencies* (*negative* $\lambda(a)$). They reflect the fact that a set of portfolio elements jointly released carries *less* information than the composition of the labels of the individual elements. For instance, the association between a complete `Address` and `Country` can be considered less sensitive than the sum of the sensitivity labels of the two. As a matter of fact, the information carried by the address includes the country where the user lives.

Disclosure constraints. The user may want to specify additional constraints that cannot be simply expressed with a sensitivity label.

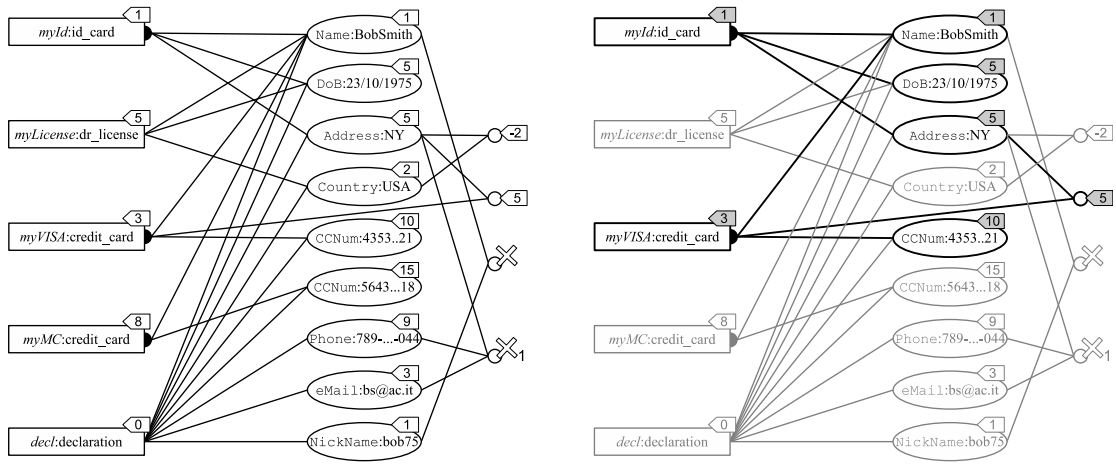


Figure 3: An example of a portfolio graph, extended with sensitivity labels, associations, and disclosure constraints (a) and of a disclosure graph (b)

- *Forbidden views.* Some associations of the portfolio elements might be not only much more sensitive than the combination of the labels of the elements, but should be definitely prohibited by the user. For instance, a user might have in its portfolio both a real `Name` and a `NickName`, each one with a sensitivity label (to be considered when the element is released), but their association should never be disclosed.

- *Disclosure limitations.* In some cases, the user might wish to put a limitation of the kind *at most n of these elements* on the release of properties and credentials. For instance, a user may specify disclosure limitation $\{$`Address`,`Phone`,`eMail`$\}_1$, meaning that at most one among its contacts can be released.

The portfolio graph is therefore extended to include both associations and disclosure constraints, which are modeled as vertexes in the graph, with edges connecting each association/disclosure constraint with the involved properties and/or credentials. Each vertex in the extended portfolio graph, but disclosure constraints, is associated with its sensitivity label. Figure 3(a) illustrates an example of a portfolio graph, obtained by extending the portfolio in Figure 2.

1.2.3 Server request and client disclosure

Consider an open web-based scenario that consists of remote communications between a client and a server. The client starts the communication by requesting access to a resource available on the server. In turn, the server sends a *request* for a set of properties to the client, to evaluate and enforce the policies protecting the resource. The request of the server is modeled as a boolean formula $\mathcal{R}$, which describes the set of properties (and

the way in which they should be certified) that the client needs to disclose to acquire a given service.

A disclosure $\mathcal{D}$ represents a subset of the client portfolio, which is communicated to the server for satisfying a request. A disclosure can be modeled as a subgraph of the portfolio graph, called *disclosure graph*. Intuitively, this subgraph includes all the vertices and edges corresponding to credentials, properties, associations, and disclosure constraints that are exposed by the disclosure. Note that the disclosure to the server of a subset of the properties in the portfolio must also imply the release of a set of credentials certifying them, additional properties included in atomic credentials, associations, and disclosure constraints. Therefore, while each disclosure is a subgraph, the vice versa is not necessarily true. A subgraph of the portfolio graph can be considered a disclosure graph only if it correctly represents a possible release of information. In particular, in a disclosure: *1)* each disclosed property must be certified by (at least) a credential, that is, credential existence is also disclosed (*certifiability*); *2)* if a property of an atomic credential is disclosed, all its properties are disclosed (*atomicity*); *3)* if all properties/credentials composing an association are disclosed, the sensitive association must be considered disclosed (*association exposure*); *4)* if all properties/credentials composing a forbidden view or more than n properties/credentials composing a disclosure limitation are released, the disclosure constraint must be considered violated (*constraint violation*). Figure 3(b) illustrates an example of a disclosure graph, which is a subgraph of the portfolio graph in Figure 3(a). The vertices and edges in the portfolio graph that also belong to the disclosure graph are represented with a bold line in the figure.

The sensitivity label $\lambda(\mathcal{D})$ of a disclosure $\mathcal{D}$ is computed by composing the labels of the credentials and properties in $\mathcal{D}$, and of the exposed associations. A disclosure is said to be *valid* if it does not violate any disclosure constraint specified by the client. Note that only valid disclosures can be released. Figure 3(b) illustrates an example of a valid disclosure $\mathcal{D}$, with $\lambda(\mathcal{D}) = 1 + 5 + 5 + 10 + 1 + 3 + 5 = 30$.

Given a server request $\mathcal{R}$, the client is interested in determining, if it exists, a valid disclosure $\mathcal{D}$ that satisfies $\mathcal{R}$, while minimizing the sensitivity label of the disclosure. This problem is however NP-hard (the minimum cover problem reduces to it in polynomial time) [ADF+10d].

1.2.4 Computing a minimal disclosure

Since the problem of computing a valid disclosure $\mathcal{D}$ that satisfies $\mathcal{R}$ with minimum sensitivity label is NP-hard, it is necessary to design efficient solutions able to compute a good, although non optimal, solution to the problem. To this purpose, two different approaches can be adopted, as described in the following.

Graph-based approach [ADF+10c]. This solution is based on a graph modeling of the server request and uses graph isomorphisms to check if a disclosure graph satisfies a request. Even if the problem of computing a minimal disclosure that satisfies a request resembles a problem of graph matching, the model illustrated [ADF+10c] has some peculiarities that cannot be simply handled by off-the-shelf graph matching algorithms. As a consequence, a specific heuristic algorithm has been designed to take the peculiarities of the problem into consideration.

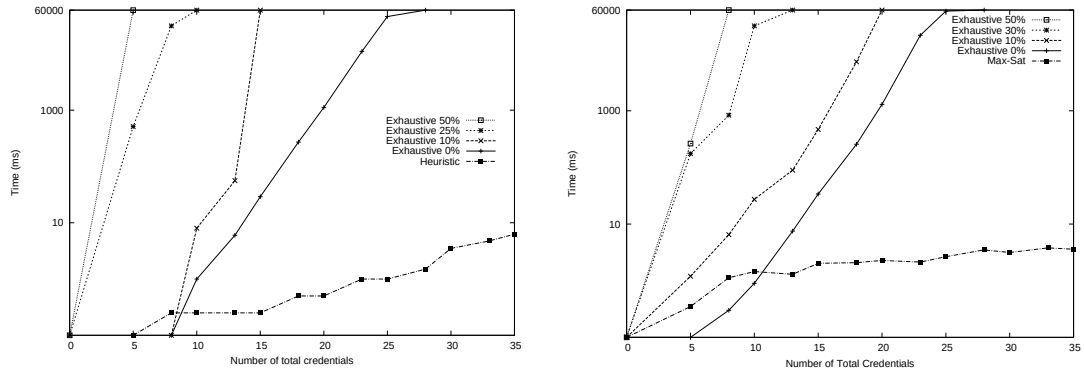


Figure 4: Execution time of the heuristic algorithms (a) and the Weighted Max-SAT prototype (b), compared with the exhaustive algorithm

To assess the efficiency and effectiveness of the heuristic, the algorithm has been implemented in C++. Experiments have been run on a PC with two Intel Xeon Quad 2.0GHz L3-4MB, 12GB RAM, and a Linux Ubuntu 9.04 operating system. Overall, the heuristic algorithm was able to produce the optimum in 98% of the cases, and when the optimum was not identified, the distance from the optimum was on average 13% above the optimum. Figure 4(a) compares the execution time of the heuristic with the execution time of an exhaustive algorithm, considering an increasing number of credentials (from 0 to 35) and 4 configurations obtained by assuming 50%, 25%, 10%, and 0% of the credentials to be non-atomic. As expected, the heuristic was always able to produce an answer in less than 10ms, whereas the exhaustive algorithm requires exponential time in the size of the portfolio, with a strong dependence on the number of non-atomic credentials.

SAT-based approach [ADF$^+$10d]. This solution is based on the translation of the problem of computing a minimal disclosure into an instance of the *Weighted Max-SAT* problem and on the availability of SAT solvers, known to be able to efficiently solve large problems. This translation interprets credentials and properties as Boolean variables, and associations, disclosure constraints, and the server request as clauses. The Weighted Max-SAT problem can be formulated as follows: *given a set of clauses, find a truth assignment that maximizes the sum of weights of satisfied clauses.*
To prove the effectiveness and evaluate the efficiency of the solution proposed, a prototype that realizes the translation of any instance of the problem into an equivalent instance of the Weighted Max-SAT problem has been realized. The set of clauses obtained by the prototype are then given as input to the Yices SAT solver (http://yices.csl.sri.com). Experiments have been run on a PC with two Intel Xeon Quad 2.0GHz L3-4MB processors, 12GB RAM, and a Linux Ubuntu 9.04 operating system, considering four different portfolio configurations, where 50%, 25%, 10%, and 0% of the credentials are non-atomic. Figure 4(b) compares the performance of the proposed solution with the exhaustive algorithm. As expected, the exhaustive algorithm becomes quickly infeasible, with a strong

dependence on the number of non-atomic credentials. On the contrary, the approach that uses the Max-SAT solver can compute, for all the considered configurations, a solution in less than 10ms.

1.3 Policy definition and management in open scenarios

Traditional access control solutions, based on preliminary identification and authentication of the access requester, are not adequate for the context of open web service systems, where servers generally do not have prior knowledge of the requesters. The research community has acknowledged such a paradigm shift and several investigations have been carried out for new approaches to regulate access control in open dynamic settings. Many works and progresses in credential-based and attribute-based access control rely on the idea that the server communicates to the client the credentials that she must possess, or the properties that she needs to satisfy, to acquire access. Several works have also investigated the different aspects of credential-based access control, and presented different models and languages (e.g., [ACK+10, ADN+10, BS02, IY05, JSSS01, WCJS97, YWS03]). Typically, a logic-based language is proposed, allowing for compact and expressive specifications of the access control policy as well as its communication to the client. Furthermore, when privacy is an issue, these works assume that the client can enforce her own policy and initiate a negotiation with the server, during which access policies, credentials, and attributes are exchanged, until access is eventually granted or denied. Nevertheless, many logic-based proposals, while appealing for their expressiveness, turn out to be not applicable in practice, where simplicity, efficiency, and consistency with consolidated technology are crucial. This section describes novel approaches that effectively tackle these issues to provide enhanced access control systems that address the requirements of open scenarios.

1.3.1 Expressive and deployable access control policies

Access control systems that fit open and distributed scenarios need to consider novel concepts that include the support for: *i)* certified information to express that some properties should be presented by means of given certificates, possibly imposing conditions, besides on values of the properties, also on the certificates themselves (e.g., their type or issuer); *ii)* abstractions representing shorthands to express, with a single concept, a more complex definition (e.g., a set, a disjunction, or a conjunction of concepts); *iii)* recursive reasoning on credentials or their properties for expressing policies based on chains of credentials and for supporting delegation; *iv)* dialog management, that is, a new way of enforcing the access control process, which cannot be assumed anymore to operate with a given prior knowledge and return a definite access decision.

A formal description of the above novel concepts is presented, focusing on the formal representation of the basic building blocks needed for referring to credentials and reasoning about them, and for supporting abstractions, recursion, and dialog [ADP+11].

Credentials

Credential support requires the possibility of explicitly referring to digital certificates and relevant conditions about them in the policy specifications. A credential is formally modeled as follows.

Definition 1.3.1 (Credential) A credential is represented by a symbol c and is modeled as a pair ($\mathcal{M}_C$,$\mathcal{A}_C$), where:

- $\mathcal{M}_C$ is a list of *metadata* name-value pairs ($\langle M_1, m_1\rangle, \ldots, \langle M_k, m_k\rangle$) that represent properties on the credential (e.g., $\langle$*type*,id_card$\rangle$ is a metadata describing a credential of *type* id_card);
- $\mathcal{A}_C$ is a list of attribute name-value pairs ($\langle A_1, a_1\rangle, \ldots, \langle A_n, a_n\rangle$) that represent the *content* of the credential (e.g., $\langle$*last_name*,Smith$\rangle$ represents attribute *last_name* whose value is Smith).

The *schema* of a credential c, denoted $\underline{c}$, is the set of its metadata and attribute names alone, without the specific instances of values. Formally, $\underline{c} = (\underline{\mathcal{M}_C}, \underline{\mathcal{A}_C})$, where $\underline{\mathcal{M}_C} = (M_1, \ldots, M_k)$, and $\underline{\mathcal{A}_C} = (A_1, \ldots, A_n)$. The proposed notation relies on the triangle ($\triangleright$) symbol and the dot (.) symbol to create *credential terms* that refer to metadata and attribute names, respectively, in a credential schema. For instance, terms $c \triangleright M$ and $c.A$ refer to metadata name M and to attribute name A in credential c, respectively.

Credential terms can then be used to specify *credential conditions* on certified properties and on the certificates themselves, which can be used like any other condition within a subject expression (i.e., the attribute-based expression identifying subjects to which the authorization applies).

Definition 1.3.2 (Simple credential condition) *A* simple credential condition *is either:*

- *a credential term t, where t is $c \triangleright M$ or $c.A$, with $\underline{c}$=($\underline{\mathcal{M}_C}$,$\underline{\mathcal{A}_C}$), $M \in \underline{\mathcal{M}_C}$, and $A \in \underline{\mathcal{A}_C}$;*
- *an expression of the form $t\ \pi\ v$, where t is credential term, π is a symbol representing a standard predicate (e.g., '$=$', '$\neq$', '$>$', '$\in$'), and v is a metadata value, an attribute value, or another credential term.*

Conditions of the first type (credential terms) have the semantics of requiring the client to hold a credential with the specified term. If no further condition is specified on the corresponding metadata/attribute in the subject expression, the semantics is simply that the metadata/attribute needs to be presented, although its value does not impact the access control decision. For instance, a subject expression can include as a condition a term *c.last_name*, without any further condition on attribute *last_name*; while the value of the attribute is not taken into consideration in the access control decision process, it might be needed for logging purposes (as the name of those who have accessed a service is logged).

Conditions of the second type are satisfied if the client holds a credential with a term that satisfies the stated restriction. For instance, $c.last\_name$ = Smith is satisfied by a credential including attribute *last_ name* whose value is Smith. Simple credential conditions can be combined by using the AND and OR boolean operators to create more complex conditions, as formally defined as follows.

Definition 1.3.3 (Credential condition) *A* credential condition *is inductively defined as:*

- *a simple credential condition, or*
- $s_1 \wedge s_2$, *or*
- $s_1 \vee s_2$,

where s_1 *and* s_2 *are credential conditions.*

A credential condition represents the basic construct to be used in the definition of a subject expression in a policy that permits referring to a set of subjects satisfying certain conditions. Note that a credential condition may include multiple occurrences of credential symbols. If the same symbol is used, the corresponding simple conditions refer to (and must therefore be satisfied by) the same credential; if a different symbol is used, the corresponding simple conditions can be satisfied by different credentials or must be satisfied by different credentials, if an inequality between the credential symbols is explicitly specified in the credential condition. For instance, condition "$c_1 \triangleright type$ = passport $\wedge$ $c_1.last\_name$ = Smith $\wedge$ $c_1.nationality$ = US" can be satisfied by presenting a credential of *type* passport containing attributes *last_ name* and *nationality* with values Smith and US, respectively. Condition "$c_1 \triangleright type$ = passport $\wedge$ $c_1.last\_name$ = Smith $\wedge$ $c_1.nationality$ = US $\wedge$ $c_2 \triangleright type$ = credit_card $\wedge$ $c_2.last\_name$ = Smith $\wedge$ $c_2.cc\_number$" can be satisfied by two different credentials, the first one of *type* passport, containing attributes *last_ name* and *nationality* with values Smith and US, respectively, and the second one of *type* credit_card containing attribute *last_ name* with value Smith and reporting the credit card number (note that the expression is satisfied by the presence of attribute *cc_ number*, regardless of its value).

Dealing with credentials requires distinguishing, in the language, between certified and uncertified properties (i.e., properties that can appear in a declaration). Requests for uncertified properties can be expressed by using simple uncertified conditions of the form 'A' or '$A\ \pi\ v$', where A is an attribute, π a standard predicate, and v a value or another attribute. As for credential conditions, also simple uncertified conditions can be combined by using the AND and OR boolean operators. For instance, "*last_ name* = Smith $\wedge$ *nationality* = US" is satisfied if the client declares (without any certification) that the *last_ name* is Smith and the *nationality* is US.

Abstractions

An abstraction defines a shorthand for new concepts that can be expressed in terms of conditions on other concepts. Such abstractions simplify the specification of conditional expressions and provide a support for ontological reasoning. Formally, an abstraction is defined as follows.

Definition 1.3.4 (Abstraction) *An abstraction is a rule of the form* simple_cond ← cond*, where* simple_cond *is a simple credential or uncertified condition, and* cond *is a credential or uncertified condition.*

For instance, abstractions:

- $c \triangleright type$ = id_document ← $c \triangleright type$ = identity_card ∨ $c \triangleright type$ = passport ∨ $c \triangleright type$ = driver_license;
- $c \triangleright type$ = emoney ← $c \triangleright type$ = credit_card ∨ $c \triangleright type$ = debit_card ∨ $c \triangleright type$ = paypal

define id_document and emoney as two abstract credential types corresponding to any element in the sets of credentials {identity_card, passport, driver_license} and {credit_card, debit_card, paypal}, respectively. Hence, a request for an identifying document (credential of *type* id_document) can be satisfied by providing either an identity card, a passport, or a driver license. Abstractions can be exploited for defining and organizing concepts and taxonomies without the need for hierarchical data structures like in traditional ontologies.

Relation conditions and recursive conditions

One of the most interesting features offered by logic-based policy languages is represented by the support for recursive conditions. Recursion has a crucial role in the representation of restrictions on how authorities and, more in general, trusted parties delegate the ability to issue credentials. The delegation consists of a certification of the ability of another party to produce credentials on behalf of the delegator. In large scale distributed systems with a complex architecture, delegation increases flexibility and permits the inexpensive creation of credentials, particularly in an open environment. Such systems are characterized by application requirements calling for the specification of restrictions in delegation. The support for recursion in the policy language can be applied to the expression of conditions on data with a recursive structure.

In the following, the solution supporting delegation is discussed. Let U be the set of all users that can take part in an access control process. Let $\rho \subseteq U \times U$ be a relation between elements in U. As an example to illustrate the proposed ideas, consider particular elements in U, namely, certification authorities, and a relation ρ that holds between two certification authorities u and v if and only if u has signed v's public key on a certificate, delegating to v the authority to produce credentials, certified by v, that are to be considered as certified by u. In turn, v has the possibility to delegate her power to another certification authority, so that a chain of delegation is created, whose description must be maintained in some data structure accessible by all the users that rely on the relevant certification authorities. Θ_ρ carries this information that exhaustively describes ρ, which is called the *context* of ρ, in the form of a sequence of credential-like entries corresponding to all the pairs that ρ induces in U: $\Theta_\rho = \{\theta = (\mathcal{M}_\theta, \mathcal{A}_\theta) : \underline{\mathcal{M}_\theta} = (rel), \theta \triangleright rel = \text{rho}, \underline{\mathcal{A}_\theta} = (user_1, user_2), (\theta.user_1, \theta.user_2) \in \rho\}$.

When ρ holds between u and v, that is, $(u, v) \in \rho$, there exists a $\theta \in \Theta_\rho$ such that $\theta \triangleright rel$=rho, $\theta.user_1$ = u, and $\theta.user_2$ = v. Conditions on data with a recursive structure like

the one mentioned above can be requested in an access control policy. In the example, in which ρ is a relation of delegation between certification authorities, a client trying to access a particular resource may be required by the server to show that the certification authority ca_r signing her credentials has been delegated by a particular authority ca_s preferred by the server. The policy will then include the relevant relation condition, $\theta \triangleright rel{=}\text{rho} \wedge \theta.user_1 = ca_s \wedge \theta.user_2 = ca_r$, which, in general, can be rewritten according to the following abbreviation: $\theta.rho{=}\langle u, v\rangle \leftarrow \theta \triangleright rel{=}\text{rho} \wedge \theta.user_1 = \text{u} \wedge \theta.user_2 = \text{v}$.

In this scenario, the need often arises to deal with the transitive closure of the delegation chain. Instead of setting conditions on the authority that directly delegated the one signing the client certificate, a server may be interested in ensuring that the root authority ca_{root}, the one at the very beginning of the delegation chain, is among her preferred ones. The client can prove that her ca_r is in the relation ρ^* (i.e., the transitive closure of ρ) with ca_{root} either by showing that $(ca_{root}, ca_c) \in \rho$, or by providing a chain of context entries $\theta_1, \ldots, \theta_n \in \Theta_\rho$, where ca_{root} is $user_1$ in θ_1, ca_c is $user_2$ in θ_n, and for all $1 \leq i < n$, $\theta_i.user_2 = \theta_{i+1}.user_1$, which can be abbreviated in a *recursive condition*:

$$\theta.rho^*{=}\langle u, v\rangle \leftarrow \theta.rho{=}\langle u, v\rangle \vee (\theta.rho{=}\langle u, \theta'.user_1\rangle \wedge \theta'.rho^*{=}\langle \theta'.user_2, v\rangle).$$

Dialog management

The ability to truly support an open world scenario implies enabling the servers to process requests for services coming from parties unknown a-priori. As noted, the way access control is enforced needs to change, since in general the server has not available (either in its own state or released by the client together with the request) all the information needed for evaluating access, and cannot return a definite decision. Rather, the server should be able to evaluate the policy with respect to unknown or partially known clients, and communicate to them the conditions that need to be satisfied to access the service. eXtensible Access Control Markup Language (XACML) [OAS10] does not have this capability and returns indeterminate whenever the evaluation of the access control policy cannot be reduced to a definite state (permit or deny) and there are conditions whose truth value cannot be evaluated. The main goal is to depart from the XACML scenario allowing the server to inform the client of the conditions that it needs to satisfy instead of communicating it that there are conditions that cannot be evaluated.

An important issue is how the server should communicate its access control policy to the client. For instance, suppose that an authorization imposes that attribute *nationality* should be equal to "US". Should the server communicate such a condition to the client? Or should it just inform the client that it has to state the nationality? Clearly there is no unique response to whether one option is better than the other, and which one is to be preferred depends on the specific context and information involved. It is clear that communicating the complete policy (i.e., the fact that the policy will grant access if the nationality is US) favors the privacy of the client. In fact, a client can know, before releasing credentials or information to the server, whether the release will be sufficient to acquire access to the service. In particular, a client associated with a non-US user can avoid disclosing the nationality of the user. By contrast, communicating only part of the policy favors the privacy of the server. As a matter of fact, the access control policy, and the information on which it evaluates, can be considered sensitive too and

as such needs to be protected. For instance, while the server might not mind disclosing the fact that access to a service is restricted to US citizens, it might not want to disclose other conditions (or values against which properties are evaluated) as they are considered sensitive. As an example, consider an authorization allowing access to a service to those users who work for an organization that does not appear in a Secret Black List (SBL) kept by the server. The corresponding subject expression is: $c \triangleright \mathit{type} = \text{employment} \wedge c.\mathit{employer} \notin \text{SBL}$. Communicating the complete policy to the client (and allowing its evaluation by the client) would imply releasing the subject expression, together with the black list SBL. Also, assuming the content of SBL is not released, the client will know, in case it will not be granted access, that its employer is black listed. This is clearly an information the server does not wish to disclose; rather the server wants to maintain confidential the condition and simply state that the employment certificate is required. Between the two extremes of simply returning indeterminate (the XACML approach), on one side, and of completely disclosing the policy, on the other side, there are therefore other options offering different degrees of protection to the server policy and of information communicated to the client. Each condition appearing in the policy can then be subject to a different disclosure policy, regulating the way the presence of such a condition should be communicated to the client. Five different disclosure policies have been defined, each one potentially used independently in any condition appearing in an expression. In terms of formal notation, the disclosure policy is denoted by including the portion of a condition to not be disclosed in square brackets. For concreteness of the discussion, a condition $c \triangleright M\ \pi\ m$ is used below; the case of a condition on an attribute (i.e., of the form $c.A\ \pi\ v$ or $A\ \pi\ v$) is analogous. The following disclosure policies can be associated with the condition.

- *None.* Nothing can be disclosed about the condition. It corresponds to the XACML approach as only the information that the outcome of the policy is indeterminate is communicated, since there are conditions that cannot be evaluated. Formally, the condition will appear in the expression completely included in square brackets, that is, $[c \triangleright M\ \pi\ m]$.

- *Credential.* Only the information that there is a condition imposed on some metadata about a credential (or on some attributes of the credential) can be disclosed. The metadata (or attributes) on which the conditions are evaluated are not released. Formally, the condition will appear in the expression as $c \triangleright [M\ \pi\ m]$.

- *Property.* Only the information that a property (metadata or attributes of a credential, or uncertified statements) needs to be evaluated can be released; no information can be released on the control that will be enforced on the property. Formally, the condition will appear in the expression as $c \triangleright M\ [\pi\ m]$.

- *Predicate.* Only the information that a property (metadata or attributes of a credential, or uncertified statements) needs to be evaluated and the predicate with which it is evaluated can be released; no information can be released on the values against which the evaluation is performed. Formally, the condition will appear in the expression as $c \triangleright M\ \pi\ [m]$.

Table 2: Disclosure policies and their effect on conditions

Disclosure policy	Condition in expression	Communication to the client
none	$[c \triangleright M\ \pi\ m]$ $[c.A\ \pi\ v]$	[ ] [ ]
credential	$c \triangleright [M\ \pi\ m]$ $c.\ [A\ \pi\ v]$	$c \triangleright$ [ ] $c.$ [ ]
property	$c \triangleright M\ [\pi\ m]$ $c.A\ [\pi\ v]$	$c \triangleright M$ [ ] $c.A$ [ ]
predicate	$c \triangleright M\ \pi\ [m]$ $c.A\ \pi\ [v]$	$c \triangleright M\ \pi$ [ ] $c.A\ \pi$ [ ]
condition	$c \triangleright M\ \pi\ m$ $c.A\ \pi\ v$	$c \triangleright M\ \pi\ m$ $c.A\ \pi\ v$

- *Condition.* The condition can be fully disclosed as it is. Formally, the condition will appear in the expression with no square brackets, signaling that no component is subject to disclosure restriction, that is, $c \triangleright M\ \pi\ m$.

Table 2 summarizes the different disclosure policies reporting the formal notation with which they appear in the expression and the consequent communication to the client in the dialog.

Note that the disclosure policies of the server, affecting the information released to the client about the conditions appearing in the policy, also impact the way the client can satisfy the conditions. In particular, the credential policy implies that the client will not know which information in the credential is needed and therefore will have to release the credential in its entirety (assuming that the credential to which the condition refers is known by other conditions in the policy, else the client will have to disclose all its credentials). The property policy implies that the client can selectively disclose the property in the credential (or utter it, in case of a condition on uncertified properties). The same for the predicate policy, where the client however knows also against what predicate the property will be evaluated. Finally, in the case of the condition policy, the client can either provide the property (but it can assess, before submitting, whether such a release will satisfy the condition) or provide a proof that the property is satisfied [CL01].

Example 1.3.1 Consider a policy stating that "a user can access a service if her nationality is Italian, her city of birth is Milan, and her year of birth is earlier than 1981". Suppose that all attributes mentioned in the policy must be certified by an X.509 identity card or by a SAML passport both released by IT_Gov. The policy is formally stated as:

$((c_1 \triangleright \text{type} = \textit{identity\_card} \wedge c_1 \triangleright \text{method} = \textit{X.509}) \vee$
$(c_1 \triangleright \text{type} = \textit{passport} \wedge c_1 \triangleright \text{method} = \textit{SAML})) \wedge$
$c_1 \triangleright \text{issuer}\ [= \textit{IT\_Gov}] \wedge c_1.\text{nationality}\ [= \textit{Italian}] \wedge$
$c_1.\text{city\_of\_birth} = \textit{Milan} \wedge c_1.\text{year\_of\_birth} < [1981]$

Here, the square brackets representing the disclosure policies implicitly state that: *i)*

conditions on metadata *type* and *method*, and attribute *city_ of_ birth* can be eventually disclosed as they are; conditions on metadata *issuer* and attribute *nationality* need to be protected by hiding the control that will be enforced on them; and *iii)* condition on attribute *year_ of_ birth* needs to be protected by hiding the value against which the evaluation will be performed. If the above policy applies to a request submitted by a client for which the server has no information, the following conditions are communicated to the client.

$((c_1 \triangleright \text{type} = \textit{identity\_ card} \wedge c_1 \triangleright \text{method} = \textit{X.509}) \vee$
$(c_1 \triangleright \text{type} = \textit{passport} \wedge c_1 \triangleright \text{method} = \textit{SAML})) \wedge$
$c_1 \triangleright \text{issuer } [] \wedge c_1.\text{nationality } [] \wedge$
$c_1.\text{city\_of\_birth} = \textit{Milan} \wedge c_1.\text{year\_of\_birth} < [].$

The client can satisfy such conditions by releasing either an identity card or a passport containing the requested attributes.

Beside providing a simple and effective formalization of the novel concepts above, the work in [ADP+11] also illustrates how the modeled concepts can be deployed in the XACML standard by exploiting its extension points for the definition of new functions, and introducing a dialog management framework to enable access control interactions between web service clients and servers. In this context, it presents enhancements to be made to the standard XACML architecture to support the extended XACML language. The proposed solution consists in defining new components or in using, in a different way, existing XACML components. An important characteristic of the proposed deployment is that it has a limited impact on the original XACML specification and architecture.

1.3.2 Fine-grained disclosure of access policies

Following the proposal for dialog management in [ADP+11], Ardagna et al. [ADF+10b] present a simple, yet expressive and flexible, approach for enabling servers to specify, when defining their access control policies, if and how the policy should be communicated to the client (i.e., dialog management). The proposed approach applies to generic attribute-based access control policies and is therefore compatible with different languages, including logic-based approaches [BS03] as well as the established XACML standard [OAS10].

A policy is represented as a boolean expression tree, where operators are internal nodes, and attributes and values are leaf nodes. In the graphical representation of the policy tree, nodes representing constant values (in contrast to attributes or operators) are represented with a square.

Disclosure policy specification. The main goal of the proposed approach is to provide the server with a means to regulate how its access control policies should be communicated to clients. In fact, the server might consider its access control policy, or part of it, as confidential. To provide maximum flexibility and expressiveness, a fine-grained approach is defined where each term and predicate operator appearing in a condition, as well as each boolean operator combining different conditions, can be subject to a *disclosure policy* that regulates how the term, predicate operator, or boolean operator

should be protected and then communicated to the client. In other words, each node in the policy tree can be associated with a disclosure policy regulating if and how the existence of the node and its label should be visible to the client. With respect to the label, a disclosure policy can state whether the label of a node can be disclosed. With respect to the existence of a node and therefore the structure of the tree, a disclosure policy can state whether the structure has to be preserved or can be possibly obfuscated by removing nodes from the tree.

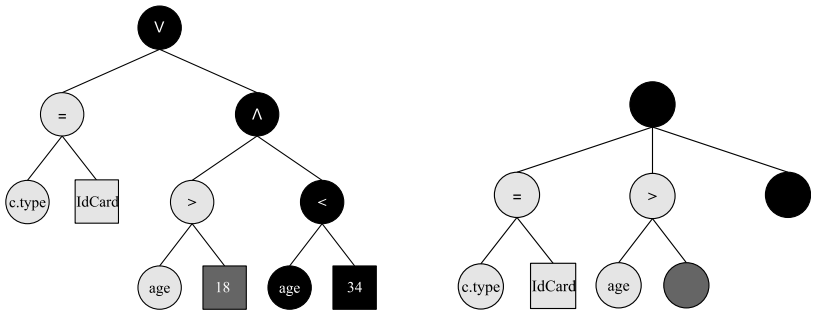
The disclosure policy associated with each node in the tree is expressed as a color (green, yellow, or red), which in Figure 5 corresponds to the light gray (green), dark grey (yellow), and black (red). The specification of disclosure policies results in a *colored policy tree*. The semantics of the different colors, with respect to the release to the client of the colored node, is as follows.

- *Green.* A green node is released as it is.
- *Yellow.* A yellow node is obfuscated by only removing its label; the presence of the node as well as of its children is preserved.
- *Red.* A red node is obfuscated by removing its label and possibly its presence in the tree.

Also, although in principle a node in a policy tree can be arbitrarily colored (i.e., any disclosure policy can be associated with any node in a tree), not all the colorings of a policy tree are *well defined*. A coloring function is well defined if all the following conditions are satisfied.

- For each green leaf representing a constant value, its sibling representing an attribute is green, and its parent, representing a predicate operator, is not red. The reason for this constraint is to not allow cases where the only information releasable to the client is the constant value against which an attribute is compared, without releasing neither the attribute nor the predicate operator.
- Each green node representing a predicate operator must have at least a non red child. The reason for this constraint is analogous to the one above. In fact, releasing a predicate operator (e.g., $>$, $<$, $=$) without its operands would be meaningless.
- For each subtree representing a basic condition on attribute `type` (of the form `c.type`=*value*), the nodes in the subtree are either all green or all red. The reason for this constraint is to ensure that if the information that there is a condition on the type of credential in the policy is released to the client, also the specific type of credential is disclosed. In fact, it would be meaningless to state that only credentials of a given type are accepted, without disclosing the type.

Apart from the constraints above restricting the diversity of colors within basic conditions of the policy, any color can be assigned to the different nodes of a policy tree, each producing a different way in which the server may wish to communicate its policy to the client. In other words, each coloring produces a possible view of the client on the policy tree.



(a) (b)

Figure 5: An example of colored policy tree (a) and corresponding policy tree view (b)

The transformation process. The colored policy tree must be transformed into an equivalent *client policy tree view*, which is then communicated to the client. The client policy tree view is obtained by applying transformation rules that: *i)* remove the label of yellow and red nodes, *ii)* remove unnecessary red leaves, and *iii)* collapse red internal nodes in a parent-child relationship into a single red node. These transformation rules are classified as: *i) prune*, prune rules operate on internal nodes whose children are leaf nodes and remove unnecessary red leaves, if any; *ii) collapse*, a single collapse rule operates on internal red nodes and removes their non-leaf red children, if any; *iii) hide label*, hide label rules operate on yellow and red nodes by removing their labels.

The transformation process works by traversing the colored policy tree with a post-order visit and, at each node, applies the rules in the order prune, collapse, and hide label. Figure 5 shows an example of policy transformation. In particular, condition (`c.type`=*IdCard*) is preserved since it is composed of green nodes only. In the subtree representing condition (`age`>*18*), the label of the yellow node *18* is removed (the structure of the subtree is instead preserved, since both node `age` and node > are green). Note that the node representing value *18* is transformed into a circle node, to hide the fact that the original node represents a value. The subtree representing condition (`age`<*34*) is pruned, since it contains only red nodes. The visit then proceeds with red node ∧ that cannot be collapsed with the red leaf node resulting from the pruning of subtree representing (`age`<*34*). Finally, the root node is visited and the collapse rule is applied, since the root has a red non-leaf child. Figure 5(b) illustrates the resulting policy view, where the red leaf represents a suppressed condition.

Properties of the client policy view. Given a policy tree view over a colored policy tree, it is important to identify the properties that characterize the policy tree view that needs to be shown to the client. These properties are useful for the server to decide whether the policy view is meaningful for the client or the application of the disclosure policies results in a view that does not represent the original policy in a "fair way". First of

all, a policy view is considered a *fair view* if it requires the minimum information needed for evaluating the corresponding policy. In addition to a fair view, a policy tree view can be characterized as *not-fair*, *over-requesting*, or *pre-evaluable*. A not-fair policy view means that the policy tree view has been obfuscated by removing too much information of the original colored policy tree. In this case, if the client releases the information requested, the policy cannot be evaluated, since some information is missing. An over-requesting policy view means that the policy tree view requires more information than the minimum information needed for evaluating the corresponding policy. This happens, for example, when the label of a disjunction node has been obfuscated by coloring it in red/yellow. A view is said to be *pre-evaluable* if the policy evaluates to true at the client side, it will evaluate to true at the server side.

1.4 Conclusions

The chapter presented some proposals aimed at supporting privacy-enhanced interactions between client and servers in open scenarios. The chapter first focused on client-side issues describing solutions for regulating the disclosure of sensitive information. It then discussed the problems on the server side illustrating approaches for defining enhanced policy languages supporting credentials, recursive condition, and fine-grained dialog management.

The research on these topics will continue after the end of the project, investigating the open issues that still need to be addressed, such as: the definition of simple while effective model helping users to associate sensitivity labels with portfolio elements based on their preferences; the support for anonymous credentials in the definition of server-side disclosure policies; and the integration of client-side and server-side solutions in a single framework.

Chapter 2

Policies for service composition (Task 5.2.2)

Task 5.2.2 focused on privacy mismatches, obligations, and privacy policies in SOA. The task analyzed the privacy mismatches that may arise when multiple layers of abstraction exist to represent privacy preferences and privacy policies. The task also studied the dependencies between authorizations and obligations with the goal of avoiding ambiguous interactions between them. Finally, the task defined an abstract framework to reason on privacy policies in Service-Oriented Architectures (SOAs).

2.1 Propagating privacy mismatches through multiple abstraction layers

When multiple representations exist for one language (e.g., XML) human readable assertions, graphical abstraction, simplified version (predefined choices, icons, etc.), it is necessary to keep track of translations to bring back feedback in the right representation. In this section, we summarize work on multiple representations of privacy policies and preferences an how mismatches are presented in those different representations.

This section presents work on keeping track of mismatches through different abstraction layers and proposing solutions to users at the right abstraction layer [Rah10]. A prototype has been implemented and allows both parties (i.e., user- and service-side policy writers) to choose their abstraction (i.e. Domain Specific Language, DSL) when specifying their privacy preferences/policies. In case of a conflict, we analyze the mismatch, iterate on possible conflicts, and display them in different abstractions by highlighting the privacy preferences and policies in the editor.

PrimeLife Policy Language (PPL) has a mechanism to measure the similarity between pieces of policies in order to precisely identify the source of mismatches. This mechanism has been implemented in Work Package 5.3 and is an indirect result of the research work presented in this section.

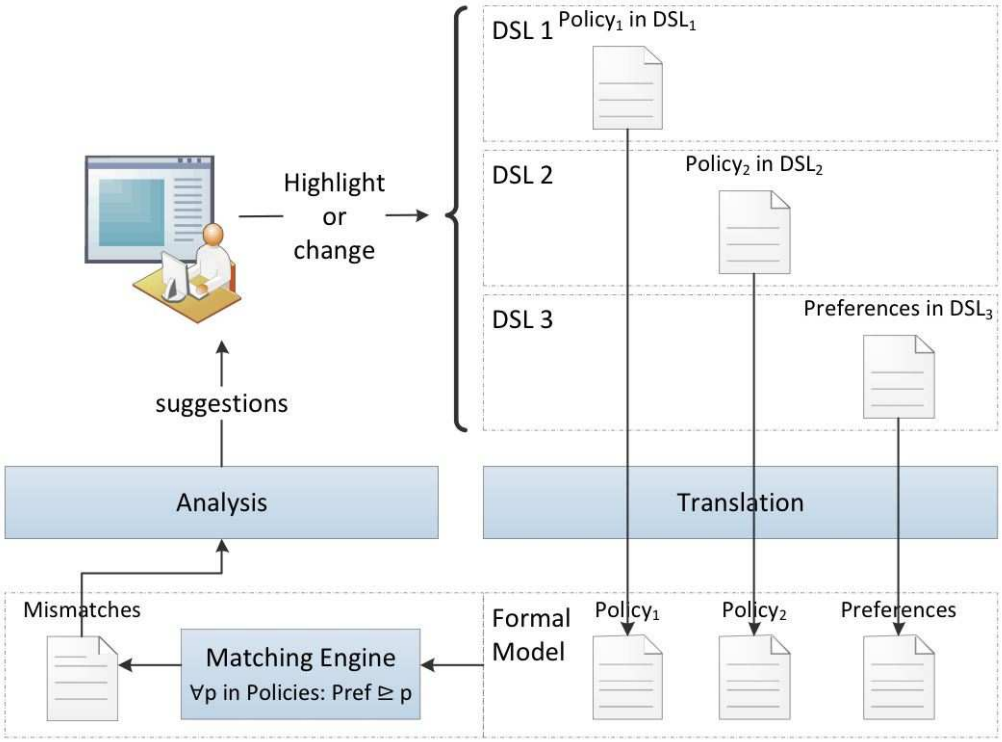


Figure 6: Overview

2.1.1 Overview

Figure 6 gives an overview of the mechanisms used to handle different DSLs. Privacy policies (e.g., front-end and downstream policies) are expressed in one or more DSLs (DSL 1 and DSL 2 in Figure 6). Privacy preferences are also expressed in a DSL (DSL 3). In the prototype, all DSLs are textual but this could be easily generalized to handle graphical DSLs as well. Each DSL covers a subset of what the formal model can express. It is thus possible to translate each DSL to the model. Note that the prototype focused on one language (i.e., PPL) but this approach could be used to combine multiple languages (e.g., PPL and P3P). This would require that the formal model cover a superset of what can be expressed with each language. In the prototype, links between assertions in the DSLs and facts in the formal model are maintained to associate results of the engine (especially mismatches) with the right part of the policies and preferences.

The prototype uses the "Formal Modeling Using Logic Programming and Analysis" (FORMULA) [JSS08] to express and reason on the formal model. The DSLs are subset of PrimeLife Policy Language (PPL) as described in [BNP10], that is, PPL where authorization is based on RBAC instead of XACML.

When the privacy policy of a service does not match user's privacy preferences, the matching engine provides a description of mismatches that is used to highlight mis-

matching parts of the policy and preferences. This information is first used to understand precisely the mismatch and, when suitable, to modify the preferences (or policy) in order to have a match. Multiple options can be proposed by the matching engine.

Matching and analysis make it possible to highlight mismatches in policies and preferences expressed with one or more DSLs and to suggest modifications in order to have a match.

2.1.2 Results and future work

This work is a first step to address a long-term goal of empowering end-users with usable and effective tools. It introduces user-friendly DSL for privacy documents, automates comparison of policy statements to each other and, mechanism guiding end-user in case of mismatch.

As the Internet continues to evolve, new privacy enabling technologies need to come up to make it a safer place for transaction and to share personal information. Users need to trust third parties (i.e., data controllers) when providing sensitive data. Otherwise the growth in e-commerce would be slowing because of consumer unwillingness to supply information. Similar discomfort also exists from organization point of view. Organizations have semi-automated procedures to handle collected data and it is often difficult to verifying whether privacy policies are indeed enforced. This work is intended to provide usable tools for end-users in both sides, to create and manage their privacy policies. Suppose, all services are bound to write their privacy practices in near future. This would require their business customers specifying their expectations as well. Both parties need a convenient tool to define their practices. They need help finding conflicts early and edit the policies if required. Writing in a natural way and getting guided assistance for detecting mismatch are critical to them.

We gathered related concerns from various aspects and advocated a set of guidelines which existing approaches need to consider for better usability and effectiveness. We further developed a proof-of-concept prototype on proposed design. We allow both parties (i.e., user and service side policy writer) to specify their privacy in DSL. In case of a conflict, we analyze our reasoning model and filter all possible conflicts at granular level, which, if resolved, would lead to a match. We not only make a report of the mismatches, but also relate them with the end-user's abstraction level (e.g., highlighting in the policy editor). We also consider priority of suggestions and sort them as the policy writer commands.

We suggest transforming the policy DSL in an intermediate structured representation. This helps achieving other important usability supports. This representation can:

1. be transformed into other more-expressive policy language(s) (we translate the DSL into PPL) which may be used either to get a structured visualization of the document or any other graphical DSL could be built on top of it. This also enables reusing a legacy system which uses that structured language on top of it.

2. be transformed into a formal model (the DSL is translated into FORMULA) that could is used to reason on preferences and policies and analyze mismatch reasons.

3. link the DSL with the implementable policy. In this way, the organizations can make sure what they publish as plain DSL is indeed what they would enforce.

4. maintain the link between different representations by, for instance, keeping track of which part of the DSL (e.g., line and position) is associated with a given fact in the formal model. This makes it possible to attach the lower-level mismatch information back to higher level DSL. Thus, the user gets feedback about the conflicts at the expected abstraction level.

Our approach of transforming the policy DSL into intermediate representation, for linking various layers as well as supporting multiple DSLs and other reasoners, is not specific for privacy domain only, rather could be generalized for other problem domains. Moreover, we allow a high level DSL on top of FORMULA which is a contribution in the area of improving support for lower level DSL.

We translated the policy DSLs in a reasoning model. In case of a non-conformity (i.e., mismatch of privacy documents) our FORMULA engine does not provide enough besides to extract exact mismatch reason(s). We address this problem by incorporating additional rules in the FORMULA domain to generate more detail knowledge about non-conformity (i.e., mismatch). In case of a mismatch, our tool extracts this additional knowledge. However, this detail information, if not re-mapped with upper level representation, would have no clue to effectively guide the end-user. Our approach, while translating to next lower level, does not loose mapping information. We process the mismatch details, associate the reasons with mapping values, and finally highlight the conflicts at the end-user abstraction.

We believe that the proposed approach, if taken forward, would open up possibilities of future work towards user-controlled privacy. From the overall perspective, there are open research problems. We summarize some issues that future research and product groups may address before our work could contribute to a generally useful privacy technology.

- *Downstream Support.* An important aspect of *usage control* (how the data has to be treated by data collector after it is released) for privacy is *downstream usage*, that is, with whom and under which usage control restrictions data can be shared [BNP10]. This is critical in the composed web service (so called *mash-ups*) scenario where the primary data controller shares user's data with other parties.

- *Model Finding.* We worked for discovering the conflict(s) behind a mismatch and highlighting accordingly. However, the policy writer needs guidance for a solution as well. We highlight conflicts in both side so that the user can get an idea with what values the conflicts need to be solved, but this is not sufficient in many cases, e.g. for missing values where the counterpart value is not specified or where the other party's policy is not visible. Better guidance needs to rely on *model finding* for generating a solution (i.e., suggestion to get a match). The reasoner we used, FORMULA, enables model finding by executing *partial models* (i.e., containing unbound but maybe restricted variables) [JSS08]. By symbolically executing such models FORMULA can find valid bindings for these variables. Using this support, our approach can be extended to automatically search for alternative solutions and provide further guidance thereby.

- *Multiple Facets.* Users may augment their preferences for other *facet*s, for example, for Service Level Agreement (SLA). Considering different facets is critical for

an end-to-end privacy management system. For instance, the user may specify her privacy as well as the service quality preference and the system needs find a matching solution considering multiple dimensions. Our work can be extended to address such scenario as FORMULA supports *separation of concern.*

- *User Interface Enhancement.* User Interface enables end-users to visualize matches and mismatches in order to make well-informed decision about sharing personal data. However, UI designers need enough information from the reasoner to enhance the usability. One of the intentions behind this work is to generate as detail information as possible behind a matching scenario. This way, our work also lays a foundation for further UI research.

2.2 Dependencies between authorizations and obligations

Usage control policies specify how the data is to be treated *after* it is revealed to a data controller. Usage control restrictions can be further divided in authorizations and obligations, where:

- an *authorization* is the right to perform a certain action on the data (e.g., forwarding the data to selected business partners). Executing an action that is not explicitly authorized by the policy is a violation of the policy. Not executing an authorized action, however, does not violate the policy.
- an *obligation* is the duty to perform a certain action (e.g., deleting the data after a certain amount of time). Not executing an obligation imposed by the policy is a violation of the policy.

While the above definition may give the impression that authorizations and obligations are separate, orthogonal parts of a policy, in [BNS10] we investigated a number of subtle dependencies that can arise between them.

1. An obligation to perform a certain action can often be rephrased as the authorization to perform a complementary action. For instance, the obligation to delete the data within one month could also be expressed as the authorization to store the data for at most one month. Vice versa, an authorization for a certain action can be seen as the obligation to not perform any complementary actions. For example, the authorization to use data for a specific purpose is equivalent to the obligation not to use the data for any other purposes.

   In previous work [BNP10] as well as in the PrimeLife Policy Language (PPL) [Pri09a], we avoided this issue by strictly separating the vocabularies for authorizations and obligations, so that no action defined in one vocabulary has a complement in the other vocabulary. SecPAL for Privacy [BMB10], on the other hand, does not separate vocabularies and requires each obligation to be explicitly authorized.

2. The execution of an authorized action may trigger the execution of an obligation. For instance, a policy could state that each access to the data for a particular purpose (authorization) needs to be logged (obligation).

3. Adhering to an obligation requires that the data controller is also authorized to do so. For instance, an obligation to notify the data subject when the data is accessed requires that the data controller is actually allowed to contact the data subject.

Our work [BNS10] focuses on the latter dependency between authorizations and obligations. In particular, we give a legal perspective and a technical solution to prevent over-diligent data controllers from "overdoing" their obligations to the extent that they become a nuisance to the data subject or degrade the quality of service. For instance, a data subject who insists to be emailed access reports for its data at least once per year may experience daily emails as spam. Worse even, the exaggerated enforcement of the obligation largely defeats its original purpose of giving the data subject an overview of the accesses to her data.

2.2.1 Legal perspective

A standard use case where a data controller has to adhere to certain obligations is credit scoring, where a scoring company is provided with a set of relevant information on loans, credit cards and bank accounts in a regular basis by the providers of the latter. Based on this data it calculates probabilities and risks of nonpayment, based on its statistical empirics.

Obviously erroneous data in this data set can have a high impact on the data subject therefore the correctness is of high relevancy. An easy approach for reducing such errors would be to regularly inform the data subject on changes in the data set. Depending on the activity of the data subject, these changes could appear quite often, which yields the risks, that the data subject would loose track of the changes and would ignore it, similar to spam. It may therefore in this case be sensible to attach an obligation that the notifications be sent on a regular basis, but not too often.

This processing calls for a legal basis, as this again is processing of personal data, with another specific purpose. One legal basis could be the law including a legal obligation for informing data subjects [Bie10]. Another option could be contractual, where a credit scoring company could derive market advantage by ensuring the data subjects be informed about the information stored by them. A number of related problems, however, cannot be discussed herein. For instance, what if an individual does not want to be informed? Might she have a "right to not know"? In the case of the results of health exams, such rights have been constructed previously [TNAAP95].

2.2.2 Technical solution

In the PrimeLife Policy Language (PPL) a data controller can specify its data handling policy *Pol*, describing how it intends to treat personal data item after it is received, and a data subject can specify her preferences *Pref*, describing how she expects her personally to be treated after it is transmitted. Both consist of authorizations and obligations. Two individual authorizations or obligations can be compared against each other by defining a partial order "more permissive than" ($\trianglerighteq$) over the vocabularies of authorizations and obligations, respectively.

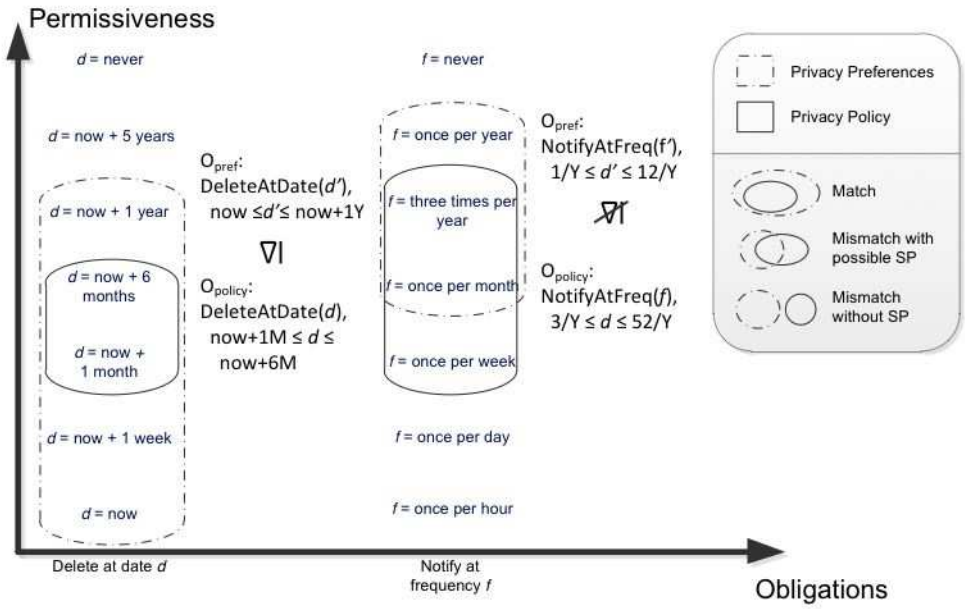


Figure 7: Examples of matching and mismatching obligations

Matching individual authorizations or obligations usually involves comparing parameter values. For instance, we have that

$$\begin{aligned} \mathit{UseForPurpose}(P) \trianglerighteq \mathit{UseForPurpose}(P') &\Leftrightarrow P \supseteq P' \\ \mathit{NotifyWithFrequency}(f) \trianglerighteq \mathit{NotifyWithFrequency}(f') &\Leftrightarrow f \leq f' \\ \mathit{DeleteWithin}(t) \trianglerighteq \mathit{DeleteWithin}(t') &\Leftrightarrow t \geq t' \ . \end{aligned}$$

A policy *Pol* is said to *match* preferences *Pref*, denoted $\mathit{Pref} \trianglerighteq \mathit{Pol}$, if and only if for all authorizations in the policy there is a more permissive authorization in the preferences, and for all obligations in the preferences there is a less permissive obligation in the policy.

PPL implicitly assumes that by proposing (imposing) an obligation in the policy (preferences), the data controller (data subject) also implicitly requests (grants) the authorization to perform the action needed to adhere to the obligation. We make the same assumption in the approach presented here.

PPL does not solve, however, the specific issue of data controllers deviating from the stated obligation parameters on the side of more privacy-friendly, but possibly more annoying values. The solution we propose is therefore that obligations, rather than specifying only the least privacy-friendly permitted value for each parameter, specify the full range of permitted values. A small range can be described simply by exhaustively enumerating its elements. If the range is a contiguous interval, it is most efficiently

represented by its endpoints. For instance, a data subject's preferences could contain the obligation

$$\mathit{NotifyWithFrequency}(\{f : \text{once per year } \leq f \leq \text{once per 3 months}\})$$

specifying that the user wants to be notified at least once per year, but at most once per three months. To match it against a corresponding obligation in the policy, one has to verify that *all* frequencies allowed by the policy are also allowed by the preferences. For instance, the obligation $\mathit{NotifyWithFrequency}(\{f : \text{once per year } \leq f \leq \text{once per 6 months}\})$ matches the preferences, but the obligation $\mathit{NotifyWithFrequency}(\{f : \text{once per 6 months } \leq f \leq \text{once per month}\})$ does not match, because the data subject does not want to be bothered by monthly reports.

Figure 7 provides a graphical representation of matching and mismatching obligations with one parameter.

In the full paper [BNS10], we present a more formal framework on how to transform any obligation with simple parameters to an obligation with intervals, and how matching should be performed on such obligations. We also observe that data controllers adhering to *more* obligations than necessary can be experienced as a nuisance, and that a similar approach affecting the matching definition of preferences against policies can be used to prevent it.

2.3 Privacy policies in Service-Oriented Architectures

In Work Packages 5.2 and 6.3, we defined an "Abstract Framework" to reason on privacy policies in Service-Oriented Architecture (SOA). This work is described in details in PrimeLife deliverable D6.3.2 [Pri11] and summarized in [BP11]. This section shows that privacy in SOA needs a lifecycle model. We formalize the lifecycle of personal data and associate privacy policies in SOA, thus generalizing privacy-friendly data handling in cross-domain service compositions. First, we describe generic patterns to enable the use of privacy policies in SOA. Second, we map existing privacy policy technologies and ongoing research work to the proposed abstraction. This highlights advantages and shortcomings of existing privacy policy technologies when applied to SOA.

2.3.1 Overview

The *Abstract Privacy Policy Framework* defines an ideal setting to enforce privacy policies in SOA. We combine experience learned in defining PPL (PrimeLife's Activity 5) and its usage in SOA (PrimeLife's WP 6.3). Different features specific to SOA are taken into account to define evaluation criteria.

- *Downstream.* Data are collected by services (data controller) that may share them with third parties. When third parties act on behalf of data controllers, they are referred to as *data processors*. When third parties are in a different trust domain, they are referred as *downstream data controllers* [BNP10]. In the latter case, the policy of third parties is taken into account by users (data subjects) when deciding whether data can provided and subsequently be shared.

- *Provider and Consumer.* Stakeholders can have both PII Provider and PII Consumer roles. For instance a service collecting customers' e-mail addresses acts as a PII Consumer but when this same service shares collected data with another (downstream) service, it acts as PII Provider.

- *Downstream accessibility.* A data subject can indirectly (i.e., downstream) interact with a third party and subsequently have a direct interaction with this one.

- *Users as PII Consumer.* In basic scenarios, users only provide data. However, a human being can also collect personal data and act as a PII Consumer. In this case the user must have a "privacy policy" expressing how she handles collected data and must accept sticky policies (e.g., license in Enterprise Right Management [Mic09]) attached to data (e.g., documents).

- *Aggregation.* Collected data can be aggregated. Mechanisms to compute the privacy constraints on the aggregation are necessary.

- *Split.* Collected data can be split. Mechanisms to compute the privacy constraints on each piece of data are necessary.

- *Privacy-aware service discovery.* Privacy impact may be taken into account when discovering services. Services may be ranked or filtered out based on privacy constraints.

- *Targeted disclosure.* PII selection (including Identity Selection) may require generating claims for a given party, or disclosing personal data to a group.

- *Distributed enforcement.* The enforcement of data handling (including access control when sharing data) is done by each party getting access to a piece of data.

- *Distributed audit.* Traces may be generated by all parties getting access to data. Mechanisms to federate the analysis of the audit traces are necessary.

The abstract framework is instantiated with concrete technologies to compare them. More precisely, criteria emerging from the abstract framework are used to compare existing privacy policy technologies and to evaluate their relevance to implement privacy policies in SOA.

The first instantiation of the abstract framework is based on a combination of two well-known standards: privacy preferences are expressed with *APPEL* (A P3P Preference Exchange Language) [W3C02] and privacy policies are expressed with *P3P* (the Platform for Privacy Preferences Project) [W3C06a]. Enforcement may rely on other technology such as *EPAL* (Enterprise Privacy Authorization Language) [AHK+03].

The second instantiation of the abstract framework is based on *PrimeLife Policy Language* (PPL) [Pri09a, Pri10] an extension of XACML [OAS10] with support for data handling. This technology is well aligned with the evaluation criteria since a large part of them were informally taken into account during its design.

The third instantiation of the abstract framework is based on *SecPAL for Privacy* (S4P) [BMB10] an extension of logic-based authorization language SecPAL [BFG10].

Logic foundations make it possible to reason on the causes of mismatches and furthermore to propose modification of preferences and/or policies thanks to abduction queries [BMD09].

The fourth instantiation refers to *remote management of access control policies* (AC) by the data subject at the data controller. In this setting, the data subject uploads her data to a data controller and configures the access control policy that must be enforced by this data controller. The evaluation assumes an expressive access control language (i.e., XACML [OAS10]). This approach can be considered as "inadequate" [KA10] to enforce privacy but is largely used. For instance *OAuth* [HL10] and *User-Managed Access* (UMA) [Kan] offer remote management of access control policies.

The fifth instantiation is based on the *PRIME Data Handling Policy* (PDH or PRIME-DHP) [ACDS08]. This language is focusing on data handling and access control but lacks important features to enable multi-hop data handling.

In this evaluation, we decided not to address technologies related to *Usage Control* and *Right Expression* such as *eXtensible rights Markup Language* (XrML) [XrM02], *Obligation Specification Language* (OSL) [PSSW08], MPEG-21 REL [Wan04], or *Open Digital Rights Language* (ODRL) [ODR02]. Even if those technologies could be used to express and enforce privacy constraints on personal data, the way constraints are agreed upon is fundamentally different than what is required to implement privacy in service-oriented architectures. Indeed, in usage control and rights management, constraints are imposed by the author (i.e., the data subject) without preliminary protocol with the party receiving the data. As a result key features such a preferences, policies, and matching algorithm are out of scope.

2.3.2 Results and future work

It appears that using P3P [W3C06a], APPEL [W3C02], and EPAL [AHK+03] together is not suitable to tackle complex scenarios. First, those technologies do not support multi-hop data handling, which is quite common in SOA. This is mainly due to the fact that those technologies are targeting Web 1.0 scenarios. Second, the use of three different languages for expressing privacy preferences, privacy policies, and their enforcement leads to semantics mismatches and difficulty to use them recursively.

Letting data subjects specify access control on their data (e.g., OAuth [HL10], UMA [Kan]) is not sufficient even when obligations can be specified (e.g., XACML [OAS10]). The main limitation is due to the fact that remote setting of access control only covers a small subset of data handling. One advantage of this approach is to limit the number of copies of personal data and to centralize their management.

PRIME-DHP [ACDS08] provides more features than P3P but does not address the preference side and complex downstream cases.

S4P [BMB10] offers promising features but only the core functionality (evaluation of queries) has been implemented. Tools for creating sticky policies, for enforcing policies, and for auditing execution traces need to be developed.

Finally, PPL [Pri09a] supports a large part of the abstract privacy policy framework. This is not surprising since PrimeLife's work packages on SOA and on Policies are strongly connected. PPL mainly lacks homogeneity and logic foundations to enable reasoning on the policies. Both issues are related to the use of XACML as underlying

authorization language. XACML is indeed very expressive and difficult to represent formally. As a result, it is not possible to enumerate different modifications of preferences and policies to resolve a mismatch.

Future work will add instantiations by evaluating the use of Usage Control and Right Expression Languages. Moreover, the number of criteria will increase by refining existing evaluation criteria. This work will also impact the evolution of PPL and other policy languages we are contributing to.

Chapter 3

Legal policy mechanisms (Task 5.2.3)

The research in Task 5.2.3 aims especially at the description and analysis of legal requirements for privacy policies, thus building the basis for technical requirements. The results shall lead to a new generation of technically supported privacy policies that implement legal requirements in a suitable way. Understanding the underlying legal mechanisms and how these match with reality and business practices is a necessary foundation for preparing privacy policies. The legal research within this task had been driven by PrimeLife partner ULD. In the third year of the project, the previous legal research has been continued and results have been disseminated towards other activities (e.g., Activity 4) as well as external stakeholders. To deploy machine readable and matchable policies it is important to understand which content is essential for policies to be compliant with legal regulations. During the research leading to H5.2.2 [HS11], but also during discussions on how the content of privacy policies could be visualized in user-friendly interfaces (Work Package 4.3), it became clear that the purposes of data processing are a central element of policies. Describing purposes is particularly necessary to provide a legal basis for any processing of personal data – be it the informed consent of the user or directly based upon the law. Purposes also play a central role for assessing which processing of data may be deemed necessary and is thus allowed by the law.

To develop this understanding further, research on legal policy mechanism aims at developing the basis for taxonomies and partonomies that can be used as a starting point in defining vocabulary for technical and legal privacy policies and languages expressing the latter. To provide a basis for the aforementioned, work on methodologies was also necessary. Generally, for developing a typology of the processing types of personal data an empirical approach, looking at the current practice, seems appropriate. The specialties, however, are subject to discussion.

Transparency is one of the core principles of data protection legislation in Europe, Art. 7 of Directive 95/46/EC [Com95], beyond [APE05] and all around the world [OEC80]. The common understanding is that individuals should be aware of

'who knows what about them' [Ger83]. This concept is supported by the principle of purpose or collection limitation. This principle stipulates that any collected personal information may only be processed for those purposes it was collected for. Roots of this principle can be seen in the sociological concept of 'functional differentiation' [Luh77], or, related, in what Nissenbaum calls 'contextual integrity' [Nis04, Nis10, Nis98]. They appear to be basic conditions of just communication and social interaction in democratic societies. Often enough, these principles are hard to enact, enforce and above all hard to understand for a user. The user is confronted with a multitude of different purposes, often hidden in the lengthy legal text of privacy policies, especially when surfing the web.

A number of approaches are currently trying to tackle this problem, by offering the user tools and mechanisms for a better understanding of what is happening with their data. However, in most if not all of these approaches it is unclear what is actually necessary to communicate. The problem relates to the above. For a higher level of transparency, the user should be made aware of what actually happens to the data, who is processing them, and – if collected without the informed consent or knowledge of the user, what data are processed (e.g., Cookies, IP addresses, clickstreams). While the types of data processed may be easy to communicate (but still might need some further thought), the question of expressing in a simple way, how the data are processed, and for what purpose(s) they are collected, poses difficulties. The multitude of applications and uses of personal data are highly unstructured, as no comprehensive ontology exists, and no abstractions are apparent.

The work in PrimeLife's Activity 5 on Next Generation Policies, especially in Task 5.2.3 on legal policy mechanisms, aims at a better understanding of the legal aspects of the processing of personal data, by looking at the current status of this processing in different contexts and structuring these.

3.1 Legal framework for processing personal data

Recent and ongoing work in user transparency and legal privacy policies is done in the area of Human-Computer Interaction (HCI), as well as in technical representations and functional descriptions of policies for privacy and data protection. The Article 29 Working Party has endorsed the use of multi-layered legal policies for websites [Eur04], which has been implemented in several places on the web [Bro08]. Others have been developing tools and interfaces to support the user [HCI07]. Finally there are proposals to use iconography to simplify the recognition of a legal privacy policy for the user [Fis06, Run06] that have gained some momentum [Pri09b].

For a formal description of privacy policies, P3P is an available specification [W3C06a], and offers some structural reference. Approaches discussed for expressing rules include: XACML [OAS10], EPAL [W3C03], Liberty's Internet Governance Framework [Ide07], and WS-Policy [W3C06b]. However, these specifications are very limited, or even completely lacking in offering conventions on describing the aspects most relevant to the user: What is actually going to happen to the data, what purposes are they used for, under what conditions and with what obligations, and will they be passed on to third parties, if so to whom?

Last but not least the PRIME project [PRI07] has initiated some research for a more

thorough ontology in this area and has reached some first results, which further research should take into account.

General legal requirements for processing personal data are defined in the Data Protection Directive 95/46/EC and the Directive on Privacy and Electronic Communications 2002/58/EC (DPEC). Legal requirements for privacy policies encompass data collection as well as data handling, i.e. the processing. The core element of Directive 95/46/EC is only allowing the collection and processing of personal data if there is a legal basis or if the data subject unambiguously has given her consent, cf. Article 7 of Directive 95/46/EC.

The legal basis are specific or general provisions as they are contained within the national data protection laws based on transformations of Art. 7 b-f of Directive 95/46/EC. These general legal regulations usually require that the processing must be necessary for the purpose stated within the law. To the best of our knowledge, the element of necessity had not been analyzed in a broader context so far and the sighted legal literature provides examples rather than a systematic overview.

Where collecting data is permissible, the further process requires the data controller to ensure that the data is kept accurate and only used for the purposes for which they have been collected. While the former are relatively easy to define, a specific problem of the legal requirements is defining the purpose of data collecting.

The difficulty of how precise the policy language has to be, can also be seen in light of Directive 95/46/EC. According to Article 12, lit a, bullet point 3, the data subject has the right of information about the data handling. This is also 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. The data collector also needs a precise policy to comply with her promises when processing the data internally in her own business. To ensure the implementation of legal requirements during data handling, the data collector has to advise her employees on how to process the collected data. To avoid mistakes in handling data, the advice has to be as precise as possible.

3.2 Gaps in current policy language approaches

Sticky policies promise to improve the state of the art in data protection, both on the level of better control and on the level of increasing transparency. A 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. Associated with a resource, it is the agreed-upon sets of granted authorizations and promised obligations with respect to a resource. Sticky policies allow an enforcement of the policy (e.g., deletion at the end of the retention period), and thus allow the control on how data are to be accessed and used and that accompany data throughout an entire distributed system [CL08].

A privacy policy that would support privacy through sticky policies would need to implement legal requirements for the employees of the data collector as well as for the data subject. The maxim of transparency in data collecting and data handling also argues for the assumption that a privacy policy has to be as precise as possible. This

clarifies the need for a precise and well-implemented privacy policy. A number of different and "complementary" approaches are currently being taken to support legal compliance in current IT systems.

The objective of these approaches is to support compliant use of the system by mixing the appropriate technological and organizational mechanisms. The legal framework is ideally already introduced when defining the specification of the system. Numerous policy languages currently available or under development address this. In the following, two approaches, XACML (extended) and P3P, will be highlighted and analyzed with respect to their potential to support legal compliance.

3.2.1 XACML – Extensible Access Control Markup Language

EXtensible Access Control Markup language (XACML) is an XML-based language for expressing and interchanging access control policies. The language's core functionalities are geared towards access control, but it also offers standard extension points for defining new functions, data types, policy combination logic and more. In addition to the language, XACML defines both an architecture for the evaluation of policies and a communication protocol for message interchange, as well as supporting multiple subject specifications in a single policy, multi-valued attributes, conditions on metadata of the resources and policy indexing. XACML is an applicable mechanism for technically implementing legal requirements of access control. The control of data handling on the other hand is not fully covered by XACML, but may potentially be achieved by means of its extensibility.

During the development of PrimeLife, some effort has been spent on extending XACML. This work suggests a new obligation handling mechanism, taking into account temporal constraints, preobligations, conditional obligations and repeating obligations together with an authorization system, defining the access control rules under which personal information collected by an entity can be forwarded to a third party (down-stream usage) [ABD+09].

Part of this work is based on the concept of trusted credentials. XACML was extended with data handling and credential capabilities. The overall structure of XACML was maintained, and a number of new elements to support the advanced features the language offers were introduced. It can be used by the data controller as well as by the data subject. The result of the completed research towards XACML-PrimeLife is that it is a suitable way to display the party collecting data, but that there is still difficulty in displaying the purpose of the data processing. Currently, the research forms a wrapper in this regard that can be used to contain elements of the Platform for Privacy Preferences (P3P), but also could include different ontologies.

3.2.2 P3P - Platform for Privacy Preferences

For a formal description of privacy policies, P3P is an available specification that offers some structural reference [W3C06a]. P3P enables websites to express their privacy practices in a standard format that can be retrieved automatically and interpreted easily by user agents.

Concern has been raised that P3P may be too complicated to be understandable for

users. However, P3P has the advantage of describing the aspects most relevant to the user: what is actually going to happen to the data, what purposes they are used for and under which conditions and with what obligations [Rag09].

Moreover, the expressiveness 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 [W3C06a]. 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, as we have analyzed in a number of scenarios [HS11] and will need further extension in its vocabulary, potentially by way of a more comprehensive ontology, or at least a taxonomy or partonomy.

3.3 Methodology

The methods that had been applied consisted of desk research in legal literature and judicature and an analysis of use cases. To derive a basis for developing a vocabulary following the criteria specified for sticky policies, empirical approaches were analyzed and evaluated with regards to their value for developing such a basis:

- the German corpus iuris, especially those norms in federal law, allowing the processing of personal data by public bodies and agencies;
- "Verfahrensverzeichnisse", a regulation specific to German data protection law that mandates data controllers to maintain a list of those processes, within which they are processing personal data, cf. § 4 lit. e BDSG;
- privacy policies from the internet, possibly in cooperation with further entities;
- PrimeLife use cases.

The goals of such an approach are to find and to define vocabularies for the following attributes:

- processes and services for personal data;
- purposes of processes and a partonomy/taxonomy thereof, to be sorted by relevance;
- typical sets of data, expressed as a partonomy of data types and data sets;
- data types and possibly qualifications/attributes (such as sensitive information as defined by Directive 95/46/EC);
- reference to the legal basis for the processing (norms, legal privacy policies);
- text elements from legal privacy policies;

- possible further elements, especially obligations, such as logging, deletion, blocking, further information (e.g., in case of incidents), and retention periods (currently not included in the research);
- categories of data processors, and a partonomy/taxonomoy thereof (currently not included in the research).

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

3.3.1 Looking into privacy policies

Our research, conducted on a limited number of privacy policies, has shown differing results on the usability of the analyzed policies. We have analyzed 34 privacy policies, most of them from Dutch websites and therefore most of them under the legislation of the Directive 95/46/EC. From the analysis resulted that some of the privacy policies implemented the legal requirements for collecting data in a legally compliant manner while others did not. The implementation of legal requirements for data handling on the other hand was not displayed very well. The main problem in privacy policies seems to be the description of data handling. This is also a legal requirement, but only few policies implemented it. More detailed information about the analysis can be found in PrimeLife Heartbeat H5.2.2 [HS11].

Many of the analyzed policies did not have the level of transparency that they should have provided, which raised some concern, as to whether they complied with the requirements of the legal framework in place. This alone would certainly have been an interesting field for further research, but was not within scope of the conducted research.

Another difficulty that arises when analyzing such policies is a possible bias of interpretation. One approach to balance this bias would have been to have each policy analyzed by two researchers, with a third looking at those policies, where the previous results differed. This approach, although promising, was dismissed, taking into account the resources available for the research, but it should be taken into consideration for further research.

3.3.2 Looking at the law

The German corpus iuris appeared to be another interesting empirical basis. Due to the specific construction within German law, the processing of personal data by governmental agencies needs a specific legal basis, therefore a broad data set was to be expected.

After a selection of laws were analyzed, evaluation hinted towards the fact that this approach might not be the most effective. The specific language chosen in many cases would not provide results of the granularity necessary. Again, this indicates that even lawmakers do not achieve a reasonable level of transparency in their laws, which unfortunately also makes this approach inefficient.

A similar, but slightly different approach based on German law, was using German *Verfahrensverzeichnisse* (i.e., literally: processing directories). This specificity of German law mandates data controllers to describe each process wherein personal data is

processed. While this approach seemed promising, the available material from the German Verfahrensverzeichnisse was too limited to come to an effective result.

3.4 Use case analysis

Further research concentrated on researching selected use cases. On one hand, working with use cases has the disadvantage that a very comprehensive ontology does not seem to be in reach on this basis. On the other hand, there are several advantages in working with use cases. For instance, each possible scenario could be analyzed quite comprehensively and the problems in real life in connection with privacy protection can be displayed very well.

During the course of PrimeLife, the project has developed a set of use cases. Thus there was already a strong foundation to build upon. The approach promised a high level of compatibility for other research aspects within the project. One of the use cases that was analyzed and will be analyzed further is the scenario of a 'classical' online shop (which had already been a good scenario to display the problems concerning privacy protection during the PRIME project). Subsequently, the methodology was also performed on a social network site's use case. Social network sites comprise a higher degree of complexity related to purposes of data handling because many different constellations may appear and many different data controllers or processors respectively may exist. Given the complexity of social network sites, we were also able to assess the limits of the methodology.

During later parts of the research PrimeLife also closely cooperated with the ad-hoc working group of the German data protection authorities assessing specific questions of the newly introduced German electronic identity card (eID). The law regarding the new German eID provides that personal data will be transmitted only if the service provider has shown the holder of the eID an access certificate indicating inter alia the service provider's identity, the categories of personal data to be transmitted, and the purpose of the planned data processing, § 18 sec. 4 of the German Personalausweisgesetz. To attain this certificate, service providers need an authorization by a German government agency (Bundesverwaltungsamt). This authorization will be granted only for categories of data actually necessary for the intended purpose. This evaluation will be done by the Bundesverwaltungsamt and become part of the access certificate. To aid the authorities with the assessment of the necessity of data processing the German data protection authorities have contributed with a guideline paper. To ensure practical deployment, the analysis focused on use cases likely to become widespread use cases such as shopping scenarios, but included also views on eGovernment applications.

3.4.1 Online shopping

For reasons of brevity, the use cases can only be described by examples. For this chapter, age verification was chosen, as it has a number of interesting implications, then the legal intricacies from an analysis in the context of the German eID will be outlined to go into more depth. A more comprehensive overview of different aspects analyzed is illustrated in Figure 8, where the multitude and hierarchy of purposes is shown. The latter, however maybe misleading, often a definite hierarchy of purposes cannot be described.

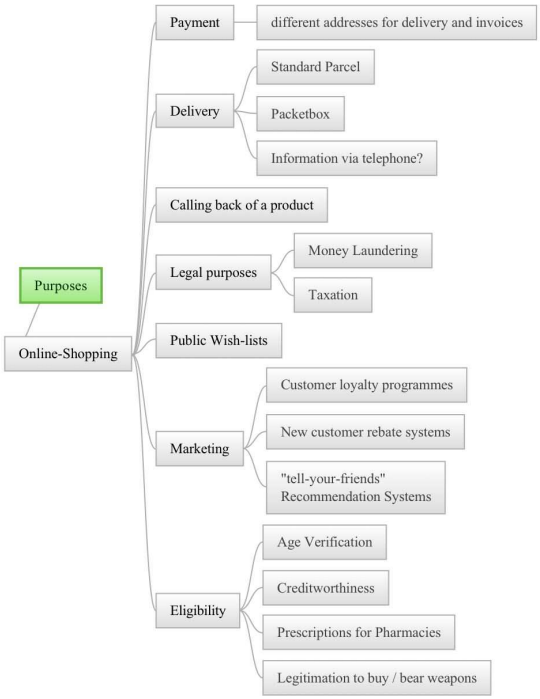


Figure 8: Overview of different purposes in an Online-shopping scenario

Coming back to the example: One of many questions for a web shop scenario is the task of age verification. If a shop - such as Amazon - sells comics like Donald Duck, there is no need to verify the age of the user. This assessment would be different where the sold good concerns (e.g., alcoholic beverages).

Often the collection may also already be legally based on the fact that it may be "necessary for the purposes of the legitimate interests" (cf. Art. 7, lit. e of Directive 95/46/EC), but in many other scenarios the controller will rely on the user actively giving consent to the processing of the information - or not be legally compliant. A broad range of all purposes for collecting data have to be displayed, to sufficiently inform the user. Paradoxically, this would render an informed decision impossible. A more thorough analysis of the web shop use case illustrates the fact that purposes of data handling can be lined out on a very granular level. From this perspective, the web shop use case appears to be a very well-defined case. Purposes can be differentiated clearly and depicted relatively easily.

Due to the research done within this task, a method for the assessment of the necessity to process personal data had been suggested that can be applied to a variety of use cases. In accordance with the principle of data minimization, it is to check whether less information may suffice. For this, a system to analyze use cases by the stages of typical phases of (contract) negotiations has been suggested. It shows that customers should be enabled to remain anonymous until they actually take over a legal obligation without fulfilling it immediately (e.g., by means of pre-payment). Exceptions apply for cases

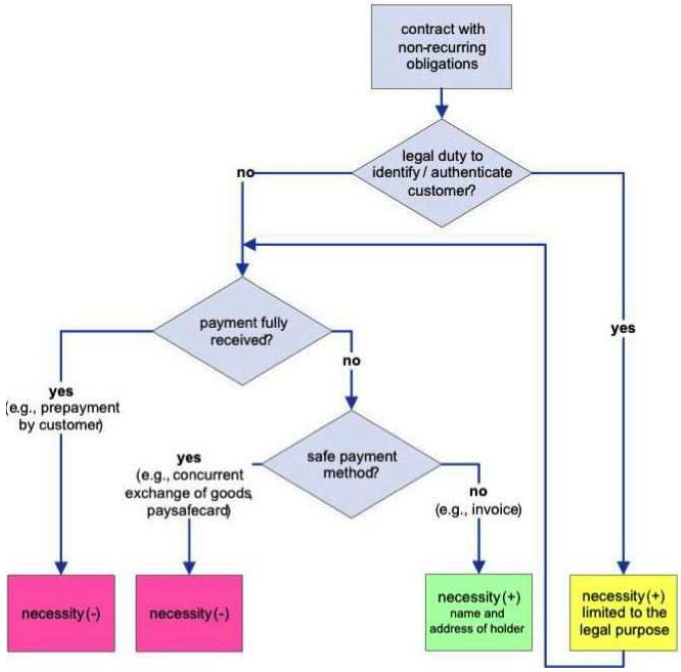


Figure 9: Necessity to process personal data for contracts with non-recurring obligations

where some kind of authorization is required because accessing a resource is limited to certain audiences such as buying alcohol or accessing adult videos or where identification is required by law as is usual in the banking sector to prevent money laundering.

This staged model had been applied to several use cases. In case of the probably most frequent transaction in practice, the exchange of goods and services for payment, an identification of the customer is necessary only when the other party bears the risk of not getting the agreed benefit in return. Only then the seller or service provider bears the risk that filing a lawsuit becomes necessary. The existence of this risk could be narrowed down to few cases where the customer is allowed to pay after having received the goods or duties in return. The decision matrix to assess can be depicted as shown in Figure 9.

Note that in the opposite direction customers of services or goods are usually allowed to identify the service provider as the law grants the customer guarantees and other rights and remedies in case the service or goods lack conformity while the rights of the service provider usually extend only to the full and timely payment. For a detailed description and other analyzed use cases see [Zwi11].

3.4.2 Social networking

Social networking implies some very specific legal questions, many of which are currently under heavy discussion within jurisprudence. They may currently be considered as one of the more difficult legal challenges in privacy legislation. It therefore seemed reasonable to

believe, that if a model could be developed for social networking, it would be possible to cover other areas with a similar approach. Over and above, better understanding privacy in social networks is one of the core research areas for PrimeLife, making studying a social network use case a logical step.

Purposes in social network sites can vary greatly. Provider-defined purposes in social network sites can be quite complex. The purposes for data handling depend inter alia on the different purposes of the social network sites per se. For instance, in those social network sites that involve primarily commercial or business objectives, other purposes may occur than in social network sites addressing leisure and recreational activities. We can therefore conclude that business oriented networks tend to define purposes for data handling mainly to promote the career benefits for the users. Other social network sites have different purposes such as "getting in touch with old classmates or friends" and "getting concrete answers to questions inside the network" [Lin10]. Such a network that is purely career oriented obviously defines other purposes for data handling. Apart from the network's business model and niche, many different functionalities can be used within social networks. For instance, Facebook offers functionalities such as mobile phone usage or SMS usage [Fac10]. The different applications again have different purposes and therefore different purposes for data handling. Thus, social network sites offer a (potentially unlimited) wide variety of user defined purposes for status messages and thereby for data handling.

On the other hand there are user-defined purposes, which can differ largely from provider-defined purposes. Users are self-defining the purposes for data handling on a case by case basis, which makes it very difficult to asses from the service provider's view beforehand. These user-defined purposes in turn also tend to be more defined by aspects of usage culture between users. In this area the research was therefore following a slightly different approach, which eventually was integrated into a separate project. One way of predefining purposes in a user-defined scenario might be the Privicon approach [CSBS], where users can predefine how the recipient of an email should handle the mail.

3.5 Outreach and potential deployment

The research had been published in German language as guidelines of the ad-hoc working party of the German data protection authorities [Dat10] and as a slightly modified guideline paper published by the Bundesverwaltungsamt [Lei10]. These guidelines are available as reference for businesses considering the application for an access certificate and are deployed by the Bundesverwaltungsamt in daily practice. An English paper has been presented at the IFIP / PrimeLife Summer School 2010 [Zwi11]. Further dissemination of the result will be done by PrimeLife Partner ULD towards the European eID community within the project SSEDIC (Scoping the Single European Digital Identity Community). Via SSEDIC various stakeholders within the European eID community can be reached. The clearer understanding of purposes and the partonoymy developed will aid data controllers to formulate privacy policies to be sufficiently accurate. This is also required from service providers applying for an access certificate with the Bundesverwaltungsamt providing potential for future outreach of these results towards the German eID community.

3.6 Central results and further research

The research completed and described within this chapter leads to the conclusion that many data controllers act on the assumption that it is precise enough to display the legitimate reason of the data collector on handling data as a legal basis (see German Federal Data Protection Law (BDSG) [Deu08], § 28, para. 1, number 2 or art. 7, lit. d of the Directive 95/46/EC) as part of their policies. This 'catch-all element' – legitimate reason – is used as a general reason (BDSG, § 28, marginal number 1). However, this should not be the state of the art of privacy policies, as it does not allow a reasonable level of transparency for the data subjects.

Our research has shown that to avoid this for the future, a more precise description of policies is necessary. Moreover, it has also demonstrated that this can be done, at least in the selected cases. The completed work includes and compares different methodological approaches, the research based on use cases as well as the research with larger empirical bases. All approaches had a common goal in mind: the need of technical policy languages, supporting privacy protection.

After carefully analyzing broader empirical approaches, a promising and extensive analysis yielded extensive efforts. To be more precise: the empirical analysis of German data protection law alone, and privacy policies of German sites only, would be immense. An analysis of published privacy policies yielded similar limitations, especially when trying to effectively rule out bias of interpretation, the effort doubles or triples. The completed empirical research also leads to the conclusion that all analyzed approaches have deficiencies. This is why a mixed approach has been taken: empirical research has been supplemented by use case-based analysis. This approach although it did not promise the comprehensiveness of an empirical analysis intended to rather focus on an exemplification of the required expressivity of the language. It includes, however, the advantage of being able to look deeper into the technical processes underlying the respective policies, rather than solely looking at what is published in privacy policies. One of the benefits of this approach is the fact that it is easier to handle a known system as a basis of research than to handle unknown systems.

Future research will have to take all possible options into account if a more comprehensive overview of data handling practices is to be achieved. A database structure for collecting such material was also developed as part of the research. The use case scenario 'online shopping' has also suggested that there are certain shortcomings in displaying data processes via the specifications that P3P offers, or at least that a higher expressivity is not only doable, but also possible. However, future research will have to look into a careful comparison of both approaches by matching the results of the use case described herein to the expressivity of P3P.

Pursuing efforts in this area of legal policy research beyond PrimeLife should include the identification of legal requirements to collect personal data of users. This collection is necessary to be able to identify where the deployment of evolving privacy preserving technologies may require adaptations of the legal context as well. It seems, for example, that many businesses collect personal data of customers primarily for the purpose of being able to file a lawsuit in case that this may become necessary. In the future this requirement could be met without storing personal data by deploying cryptographic methods. However, this may also require an adaptation of the formal requirements for

filing a lawsuit in civil procedural rules. These results may act as enablers of future eID infrastructures.

Chapter 4

Abstracts of research papers

1. C.A. Ardagna, S. De Capitani di Vimercati, S. Paraboschi, E. Pedrini, P. Samarati, M. Verdicchio, "Expressive and Deployable Access Control in Open Web Service Applications," in *IEEE Transactions on Service Computing*. [ADP+11].
   Abstract. Traditional access control solutions, based on preliminary identification and authentication of the access requester, are not adequate for the context of open Web service systems, where servers generally do not have prior knowledge of the requesters. The research community has acknowledged such a paradigm shift and several investigations have been carried out for new approaches to regulate access control in open dynamic settings. Typically based on logic, such approaches, while appealing for their expressiveness, result not applicable in practice, where simplicity, efficiency, and consistency with consolidated technology are crucial. The eXtensible Access Control Markup Language (XACML) has established itself as the emerging technological solution for controlling access in an interoperable and flexible way. Although supporting the most common policy representation mechanisms and having acquired a significant spread in the research community and the industry, XACML still suffers from some limitations which impact its ability to support actual requirements of open Web-based systems. In this paper, we provide a simple and effective formalization of novel concepts that have to be supported for enforcing the new access control paradigm needed in open scenarios, toward the aim of providing an expressive solution actually deployable with today's technology. We illustrate how the concepts of our model can be deployed in the XACML standard by exploiting its extension points for the definition of new functions, and introducing a dialog management framework to enable access control interactions between Web service clients and servers.

2. C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, P. Samarati, "Minimising Disclosure of Client Information in Credential-Based Interactions," in *International Journal of Information Privacy, Security and Integrity* [ADF+11].
   Abstract. The advancements in ICT allow people to use and access resources and services on the Web anywhere and anytime. Servers offering resources typically re-

quire users to release information about them, which is then used to enforce possible access policies on the offered services. Effective access to such resources requires the development of approaches for enabling the user to organize and manage all her credentials and regulate their release when interacting with other parties over the Web. In this paper, we provide a means for the user to specify how much she values the release of different properties, credentials, or combinations thereof as well as additional constraints that she might impose on information disclosure. Exploiting a graph modeling of the problem, the user can determine the credentials and properties to disclose to satisfy a server request while minimizing the sensitivity of the information disclosed. We develop a heuristic approach that shows execution times compatible with the requirements of interactive access to Web resources.

3. C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, T. Grandison, S. Jajodia, P. Samarati, "Access Control for Smarter Healthcare Using Policy Spaces," in *Computers & Security* [ADF+10a].
   Abstract. A fundamental requirement for the healthcare industry is that the delivery of care comes first and nothing should interfere with it. As a consequence, the access control mechanisms used in healthcare to regulate and restrict the disclosure of data are often bypassed in case of emergencies. This phenomenon, called *"break the glass"*, is a common pattern in healthcare organizations and, though quite useful and mandatory in emergency situations, from a security perspective, it represents a serious system weakness. Malicious users, in fact, can abuse the system by exploiting the break the glass principle to gain unauthorized privileges and accesses. In this paper, we propose an access control solution aimed at better regulating break the glass exceptions that occur in healthcare systems. Our solution is based on the definition of different policy spaces, a language, and a composition algebra to regulate access to patient data and to balance the rigorous nature of traditional access control systems with the "delivery of care comes first" principle.

4. C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, G. Neven, S. Paraboschi, F.-S. Preiss, P. Samarati, and M. Verdicchio, "Fine-Grained Disclosure of Access Policies," in *Proc. of the 12th International Conference on Information and Communications Security (ICICS 2010)* [ADF+10b].
   Abstract. In open scenarios, where servers may receive requests to access their services from possibly unknown clients, access control is typically based on the evaluation of (certified or uncertified) properties, that clients can present. Since assuming the client to know a-priori the properties she should present to acquire access is clearly limiting, servers should be able to respond to client requests with information on the access control policies regulating access to the requested services. In this paper, we present a simple, yet flexible and expressive, approach for allowing servers to specify *disclosure policies*, regulating if and how access control policies on services can be communicated to clients. Our approach allows fine-grain specifications, thus capturing different ways in which policies, and portions thereof, can be communicated. We also define properties that can characterize the client view of the access control policy.

5. C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, P. Samarati, "Minimizing Disclosure of Private Information in Credential-Based Interactions: A Graph-Based Approach," in *Proc. of the 2nd IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT 2010)* [ADF$^+$10c].
Abstract. We address the problem of enabling clients to regulate disclosure of their credentials and properties when interacting with servers in open scenarios. We provide a means for clients to specify the sensitivity of information in their portfolio at a fine-grain level and to determine the credentials and properties to disclose to satisfy a server request while minimizing the sensitivity of the information disclosed. Exploiting a graph modeling of the problem, we develop a heuristic approach for determining a disclosure minimizing released information, that offers execution times compatible with the requirements of interactive access to Web resources.

6. C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, P. Samarati, "Supporting Privacy Preferences in Credential-Based Interactions," in *Proc. of the 9th Workshop on Privacy in the Electronic Society (WPES 2010)* [ADF$^+$10d].
Abstract. Users can today enjoy the many benefits brought by the development and widespread adoption of Internet and related services conveniently accessing digital resources. Servers offering such resources typically require users to release information about them, which servers can then use for enforcing possible access policies on the offered services. A major problem in this context relates to providing users with the ability of determining which information to release to satisfy the server requests during their electronic interactions. In this paper, we provide an approach for empowering the user in the release of her digital portfolio based on simple sensitivity labels expressing how much the user values different properties, credentials or combinations thereof, as well as on additional constraints that the user might impose on information disclosure. We provide a generic modeling of the problem and illustrate its translation in terms of a Weighted MaxSat problem, which can be conveniently and efficiently managed by off the shelf SAT solvers, thus resulting efficient and scalable.

7. C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, P. Samarati, "Supporting User Privacy Preferences on Information Release in Open Scenarios," in *Proc. of the W3C Workshop on Privacy and Data Usage Control* [ADF$^+$10e].
Abstract. Access control solutions for open systems are typically based on the assumption that a client may adopt approached specifically designed for the server to protect the disclosure of her sensitive information. These solutions however do not consider the specific privacy requirements characterizing the client. In this paper, we put forward the idea of adopting a different model at the client-side, aimed at minimizing the amount of sensitive information released to a server. The model should be based on a formal modeling of the client portfolio and should easily support the definition of privacy preferences and disclosure limitations for empowering the user in the release of her personal information.

8. C.A. Ardagna, S. De Capitani di Vimercati, G. Neven, S. Paraboschi, F.-S. Preiss, P. Samarati, M. Verdicchio, "Enabling Privacy-Preserving Credential-Based Access Control with XACML and SAML," in *Proc. of the 3rd IEEE Interna-*

tional Symposium on Trust, Security and Privacy for Emerging Applications (TSP 2010) [ADN+10].
Abstract. In this paper we describe extensions to the access control industry standards XACML and SAML to enable privacy-preserving and credential-based access control. Rather than assuming that an enforcement point knows all the requester's attributes, our extensions allow the requester to learn which attributes have to be revealed and which conditions must be satisfied, thereby enabling to leverage the advantages of privacy-preserving technologies such as anonymous credentials. Moreover, our extensions follow a credential-based approach, i.e., attributes are regarded as being bundled together in credentials, and the policy can refer to attributes within specific credentials. In addition to defining language extensions, we also show how the XACML architecture and model of evaluating policies can be adapted to the credential-based setting, and we discuss the problems that such extensions entail.

9. L. Bussard, G. Neven, J. Schallaböck, "Data Handling: Dependencies between Authorizations and Obligations," in *Proc of the W3C Workshop on Privacy and Data Usage Control* [BNS10]
Abstract. Authorizations and obligations are keystones of data handling. On one hand there are ambiguous links between authorization and obligations. On the other hand a clear separation between both concepts is necessary to improve readability and to avoid inconsistencies. This position paper focuses on authorizations necessary to enforce obligations. Such authorizations are necessary to prevents over-diligent data controllers from "overdoing" their obligations to the extent that they become a nuisance to the data subject. This problem is discussed from a legal perspective and is addressed in a technical solution that keeps a clear separation between authorizations and obligations.

10. L. Bussard, U. Pinsdorf, "Abstract Privacy Policy Framework: Addressing Privacy Problems in SOA," in *iNetSec 2011, Open Problems in Network Security* [BP11]
Abstract. This paper wants to make the point that privacy in SOA needs a lifecycle model. We formalize the lifecycle of personal data and associated privacy policies in Service Oriented Architectures (SOA), thus generalizing privacy-friendly data handling in cross-domain service compositions. First, we describe generic patterns to enable the use of privacy policies in SOA. This summarizes our learning in two research projects: PrimeLife and SecPAL for Privacy. Second, we map existing privacy policy technologies and ongoing research work to the proposed abstraction. This highlights advantages and shortcomings of existing privacy policy technologies when applied to SOA.

11. S. Tanvir Rahman, "Analyzing Causes of Privacy Mismatches in Service Oriented Architecture," in *Master thesis at Rheinisch-Westfälische Technische Hochschule and European Microsoft Innovation Center* [Rah10]
Abstract. Internet users want controlled disclosure of their private data. They are concerned about what personal information they may reveal inadvertently while accessing websites. Intelligent systems can alleviate user's concern by assessing website's data practices automatically, assuming machine readable privacy policies.

In case of mismatch with user expectations, these systems can also help both parties reviewing their privacy statements by providing useful information. In the context of the collaborative research project PrimeLife (Privacy and Identity Management for Europe in Life), IBM, SAP, ULD, W3C and European Microsoft Innovation Center (EMIC) are working on new languages to define privacy policies. Specifying logic-based languages is important to enable reasoning on mismatches, i.e. understanding why service's privacy policy does not match user's privacy preferences. This master thesis, done with EMIC, uses domain specific language to specify privacy and focuses on mechanisms to analyze mismatches and to propose modifications for getting a match, at a higher abstraction level, e.g. DSL. In case of mismatch, this guidance permits the user judging the required amendments and make the right choice thereby, i.e. reject service's policy or modify her preference accordingly. Another concern of this work is separating different aspects of a privacy management system and link them effectively as required. The proposed approach is validated by developing a proof-of-concept prototype implementation with Microsoft's textual DSL tool, MGrammar and formal language, FORMULA.

12. H. Zwingelberg, "Necessary Processing of Personal Data: the Need-to-Know Principle and Processing Data from the new German Identity Card," in *Proc. of 6th IFIP WG 9.2, 9.6/11.7, 11.4, 11.6/PrimeLife International Summer School* [Zwi11]
Abstract. The new German electronic identity card will allow service providers to access personal data stored on the card. This imposes a new quality of data processing as these data have been governmentally verified. According to European privacy legislation any data processing must be justified in the sense that the personal data are necessary for the stipulated purpose. This need-to-know principle is a legal requirement for accessing the data stored on the eID card. This text suggests a model as basis for deriving general guidelines and aids further discussion on the question whether collecting personal data is necessary for certain business cases. Beyond the scope of the German eID card the extent and boundaries of what can be accepted as necessary data processing poses questions on a European level as well.

13. E. Forrest, J. Schallaböck, "Privicons – An Approach to Communicating Privacy Preferences between Users," in *Internet Architecture Board, Internet Privacy Workshop* [FS10]
Abstract. Alongside privacy challenges posed by technical problems like faulty architecture or insecure protocols are everyday privacy harms caused by basic failures of communication. For example, when a user unthinkingly forwards an e-mail chain that includes crass private remarks, or casually passes along information from an e-mail that was meant to have been kept secret, that user violates privacy by ignoring or misunderstanding norms, not code. Privicons uses a strategy of code-based norms or a "neighborliness" approach to address communications privacy problems like e-mail carelessness that occur within the bounds of code but neverthe- less are ultimately problems of privacy norms and social signals: problems not readily solvable by code alone. The code in Privicons thus serves mainly to help users clarify and understand social norms, signals, and expectations about privacy. If users can easily convey their privacy expectations, and recipients can

understand and process those expectations in terms of widely understood social norms about privacy, then courtesy and care will help to prevent privacy harms caused by carelessness or misunderstanding about privacy expectations.

Bibliography

[ABD+09] C.A. Ardagna, L. Bussard, S. De Capitani di Vimercati, G. Neven, E. Pedrini, S. Paraboschi, F.-S Preiss, P. Samarati, S. Trabelsi, and M. Verdicchio. Primelife policy language. In *Proc. of the W3C Workshop on Access Control Application Scenarios*, Luxembourg, November 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, September 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, January 2010.

[ADF+10a] C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, T. Grandison, S. Jajodia, and P. Samarati. Access control for smarter healthcare using policy spaces. *Computers & Security*, 29(8):848–858, November 2010.

[ADF+10b] C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, G. Neven, S. Paraboschi, F.-S. Preiss, P. Samarati, and M. Verdicchio. Fine-grained disclosure of access policies. In *Proc. of the 12th International Conference on Information and Communications Security (ICICS 2010)*, Barcelona, Spain, December 2010.

[ADF+10c] C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, and P. Samarati. Minimizing disclosure of private information in credential-based interactions: A graph-based approach. In *Proc. of the 2nd IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT 2010)*, Minneapolis, MN, USA, August 2010.

[ADF+10d] C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, and P. Samarati. Supporting privacy preferences in credential-based interactions. In *Proc. of the 9th Workshop on Privacy in the Electronic Society (WPES 2010)*, Chicago, IL, USA, October 2010.

[ADF+10e] C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, and P. Samarati. Supporting user privacy preferences on information release in open scenarios. In *Proc. of the W3C Workshop on Privacy and Data Usage Control*, Cambridge, MA, USA, October 2010.

[ADF+11] C.A. Ardagna, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, and P. Samarati. Minimising disclosure of client information in credential-based interactions. *International Journal of Information Privacy, Security and Integrity (IJIPSI)*, 2011.

[ADN+10] C.A. Ardagna, S. De Capitani di Vimercati, G. Neven, S. Paraboschi, F.-S. Preiss, P. Samarati, and M. Verdicchio. Enabling privacy-preserving credential-based access control with XACML and SAML. In *Proc. of the 3rd IEEE International Symposium on Trust, Security and Privacy for Emerging Applications (TSP 2010)*, Bradford, UK, June-July 2010.

[ADP+11] C. Ardagna, S. De Capitani di Vimercati, S. Paraboschi, E. Pedrini, P. Samarati, and M. Verdicchio. Expressive and deployable access control in open web service applications. *IEEE Transactions on Service Computing (TSC)*, 2011.

[AHK+03] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise privacy authorization language (EPAL 1.2), 2003.

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

[APE05] APEC. Chapters II and VIII of the APEC privacy framework. `http://www.apec.org/apec/newsmedia/factsheets/apecprivacyframework.MedialibDownload.v1(21.109)`, 2005.

[BFG10] M.Y. Becker, C. Fournet, and A.D. Gordon. SecPAL: Design and semantics of a decentralized authorization language. *Journal of Computer Security (JCS)*, 18(4):619–665, 2010.

[Bie10] K. Biermann. Datenbrief wird ernsthaft beraten. `http://www.zeit.de/digital/datenschutz/2010-04/datenbrief-bmi-arbeitsgruppe`, April 2010. Die Zeit Online.

[BMB10] M.Y. Becker, A. Malkis, and L. Bussard. A practical generic privacy language. In *Proc. of the 6th International Conference on Information Systems Security (ICISS 2010)*, Gandhinagar, India, December 2010.

[BMD09] M. Y. Becker, J. F. Mackay, and B. Dillaway. Abductive authorization credential gathering. In *Proc. of the 10th IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY 2009)*, London, UK, July 2009.

[BNP10] L. Bussard, G. Neven, and F.-S. Preiss. Downstream usage control. In *Proc. of the 11th IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY 2010)*, Fairfax, VA, USA, July 2010.

[BNS10] L. Bussard, G. Neven, and J. Schallaböck. Data handling: Dependencies between authorizations and obligations. In *Proc. of the W3C Workshop on Privacy and Data Usage Control*, Cambridge, MA, USA, October 2010.

[BP11] L. Bussard and U. Pinsdorf. Abstract privacy policy framework: addressing privacy problems in SOA. In *iNetSec 2011, Open Problems in Network Security*. Springer, 2011.

[Bra00] S.A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.

[Bro08] K. Brown. The infocard identity revolution. `http://technet.microsoft.com/enus/magazine/cc160966(printer).aspx`, 2008.

[BS02] P. Bonatti and P. Samarati. A unified framework for regulating service access and information release on the Web. *Journal of Computer Security (JCS)*, 10(3):241–272, 2002.

[BS03] P. Bonatti and P. Samarati. Logics for authorizations and security. In J. Chomicki, R. van der Meyden, and G. Saake, editors, *Logics for Emerging Applications of Databases*. Springer-Verlag, 2003.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proc. of the International Conference on the Theory and Application of Cryptographic Technique - Advances in Cryptology (EUROCRYPT 2001)*, Innsbruck, Austria, May 2001.

[CL08] D. Chadwick and S. Lievens. Enforcing "sticky" security policies throughout a distributed application application. In *Proc. of the 1st International Workshop on Middleware Security (Midsec 2008)*, Leuven, Belgium, December 2008.

[Com95] European Commission. Art. 7 of the data protection directive 95/46/EC, 1995.

[CSBS] R. Calo, M. Senges, A. Braendhagen, and J. Schallaböck. Privicons - privacy icons for email usage. `http://privicons.org/`.

[Dat10] Datenschutzbeauftragte. Datenschutzrechtliche leitlinien für die erteilung von berechtigungen nach § 21 abs. 2 PAuswG aus sicht der ad-hoc-arbeitsgruppe npa der datenschutzbeauftragten des bundes und der länder. Technical report, Ad-hoc-Arbeitsgruppe nPA der Datenschutzbeauftragten des Bundes und der Länder, September 2010.

[Deu08] Deutscher Bundestag. *Bundesdatenschutzgesetz in der Fassung der Bekanntmachung vom 14. January 2003 (BGBl. I S. 66), das zuletzt durch Artikel 1 des Gesetzes vom 14. August 2009 (BGBl. I S. 2814) geändert worden ist*. Deutscher Bundestag - Bundesanzeiger Verlag, 2008.

[Eur04] European Commission - Article 29 Working Party. Wp 100 opinion on more harmonised information provisions, 2004. `http://ec.europa.eu/justicehome/fsj/privacy/`.

[Fac10] Facebook profile options, 2010. http://www.facebook.com/settings/?tab=privacy\&ref=mb.

[Fis06] J. Fishenden. Creative commons and its wider potential. http://ntouk.com/?view=pli, 2006.

[FS10] E. Forrest and J. Schallaböck. Privicons - an approach to communicating privacy preferences between users. In *Proc. of the 2010 Internet Privacy Workshop (IAB 2010)*, Boston, MA, USA, December 2010.

[Ger83] German Supreme Court. BVerfGE, 65, 1 (Volkszählung, Az.1 BvR 209). *Entscheidungen des Bundesverfassungsgerichts*, 65:1, 1983.

[HBH07] D. Hardt, J. Bufu, and J. Hoyt. OpenID attribute exchange 1.0. http://openid.net/developers/specs/, December 2007.

[HCI07] HCI guidelines. https://www.primeproject.eu/primeproducts/reports/arch/pubdelD06.1.fecwp06.1v1final.pdf, 2007.

[HL10] E. Hammer-Lahav. RFC 5849: The OAuth 1.0 Protocol, 2010.

[HS11] L.-E. Holtz and J. Schallaböck. PrimeLife Heartbeat H5.2.2- Report on research on legal policy mechanisms. Internal Hearbeat within the PrimeLife Project, 2011.

[Ide07] Identity governance, 2007. http://projectliberty.org/li.

[IY05] K. Irwin and T. Yu. Preventing attribute information leakage in automated trust negotiation. In *Proc. of the 12th ACM Conference on Computer and Communications Security (CCS 2005)*, Alexandria, VA, USA, November 2005.

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

[JSSS01] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems (TODS)*, 26(2):214–260, June 2001.

[KA10] L. Kagal and H. Abelson. Access control is an inadequate framework for privacy protection. In *Proc. of the W3C Workshop on Privacy for Advanced Web APIs*, London, UK, July 2010.

[Kan] Kantara Initiative. User managed initiative. http://kantarainitiative.org/confluence/display/uma/.

[Lei10] *Leitlinie für die Vergabe von Berechtigungen für Diensteanbieter nach § 21 Abs. 2 Personalausweisgesetz*, 2010. Vergabestelle für Berechtigungszertifikate.

[Lin10] Linkedin. User agreements. http://www.linkedin.com/static?key=useragreementtrk=hbftuserag, 2010.

[Luh77] N. Luhmann. Differentiation of society. *Canadian Sociological Review*, 2:29–53, 1977.

[Mic09] Microsoft. Rights Management Services. `http://www.microsoft.com/windowsserver2008/en/us/ad-rms-overview.aspx`, 2009.

[Nis98] F.H. Nissenbaum. Protecting privacy in an information age: The problem of privacy in public. *Law and Philosophy*, 17:559–596, 1998.

[Nis04] H. F. Nissenbaum. Privacy as contextual integrity. *Washington Law Review*, 79(1), 2004.

[Nis10] H. F. Nissenbaum. Privacy in context: Technology, policy and the integrity of social life. *Stanford Law Books 65*, 2010.

[OAS10] OASIS. *eXtensible Access Control Markup Language (XACML) v3.0*, August 2010. `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`.

[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*. OECD, 1980.

[PRI07] PRIME data model. `https://www.primeproject.eu/ont/Datamodel.html`, 2007.

[Pri09a] PrimeLife Project. Draft 2nd Design for Policy Languages and Protocols (Heartbeat: H 5.3.2), July 2009.

[Pri09b] Privacy-rights-agreements. `http://wiki.igfonline.net/wiki/Privacy-rights-agreements`, 2009. Dynamic Coalition Privacy.

[Pri10] PrimeLife Consortium. Second Release of the Policy Engine (D5.3.2), September 2010.

[Pri11] PrimeLife Consortium. Infrastructure for Privacy for Life (D6.3.2), January 2011.

[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)*, Trondheim, Norway, June 2008.

[Rag09] D. Raggett. Draft 2nd design for policy languages and protocols, 2009.

[Rah10] S. T. Rahman. Analyzing causes of privacy mismatches in service oriented architecture. Master's thesis, RWTH, 2010.

[Run06] M. Rundle. International data protection and digital identity management tools (using icons to express user preferences). `http://identityproject.lse.ac.uk/mary.pdf`, 2006. Presentation at IGF2006 PrivacyWorkshop 1, Athens, Greece.

[TNAAP95] M. Temmerman, J. Ndinya-Achola, J. Ambani, and P. Piot. The right not to know HIV-test results. *Lancet*, 345(8955):969–70, Apr 1995.

[W3C02] W3C. A P3P preference exchange language 1.0 (APPEL1.0). `http://www.w3.org/TR/P3P-preferences/`, 2002.

[W3C03] W3C. Enterprise privacy authorization language (EPAL 1.2). `http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/`, 2003.

[W3C06a] W3C. The platform for privacy preferences 1.1 (P3P1.1) specification. `http://www.w3.org/TR/P3P11/`, 2006.

[W3C06b] W3C. Web services policy 1.2 - framework (WS-Policy). `http://www.w3.org/Submission/WS-Policy/`, 2006.

[Wan04] X. Wang. MPEG-21 Rights Expression Language: Enabling Interoperable Digital Rights Management. *IEEE MultiMedia*, 11(4):84–87, 2004.

[WCJS97] M. Winslett, N. Ching, V. Jones, and I. Slepchin. Assuring security and privacy for digital library transactions on the Web: Client and server security policies. In *Proc. of the 4th International Forum on Research and Technology Advances in Digital Libraries (ADL 1997)*, Washington, DC, USA, May 1997.

[XrM02] XrML 2.0 Technical Overview. XrML 2.0 Technical Overview. `http://www.xrml.org/reference/XrMLTechnicalOverviewV1.pdf`, 2002.

[YWS03] T. Yu, M. Winslett, and K.E. Seamons. Supporting structured credentials and sensitive policies trough interoperable strategies for automated trust. *ACM Transactions on Information and System Security (TISSEC)*, 6(1):1–42, February 2003.

[Zwi11] H. Zwingelberg. Necessary processing of personal data: The need-to-know principle and processing data from the new german identity card. In S. Fischer-Hübner, P. Duquenoy, M. Hansen, R. Leenes, and Ge G. Zhang, editors, *Privacy and Identity Management for Life*, volume 352 of *IFIP Advances in Information and Communication Technology*, pages 151–163, 2011.