PRIME
Privacy and Identity Management for Europe


Contract N$^o$ 507591

| | |
|---|---|
| *Title:* | *PRIME Architecture V3* |
| *Author:* | *Dieter Sommer, Marco Casassa Mont, Siani Pearson* |
| *Editors:* | *Dieter Sommer* |
| *Reviewers:* | *Marco Casassa Mont, Patrik Bichsel* |
| *Identifier:* | *D14.2.d* |
| *Type:* | *Deliverable* |
| *Version:* | *1.0* |
| *Date:* | *July 9, 2008* |
| *Status:* | *Final* |
| *Class:* | *Public* |

## Summary

This report discusses the final version of the PRIME Architecture. The architecture integrates multiple privacy enhancing technologies with the overall objective of progressing the protection of user privacy. The architecture comprises state of the art technologies such as anonymous credential systems, novel access control mechanisms, assurance assessment systems, and life cycle enforcement mechanisms.



*The PRIME project receives research funding from the Community's Sixth Framework Programme and the Swiss Federal Office for Education and Science.*

File name: D14.2.d-Architecture-V3-Final.pdf

**Members of the PRIME consortium:**

| | |
|---|---|
| IBM Belgium | Belgium |
| IBM Research GmbH | Switzerland |
| Unabhängiges Landeszentrum für Datenschutz | Germany |
| Technische Universität Dresden | Germany |
| Deutsche Lufthansa AG | Germany |
| Katholieke Universiteit Leuven | Belgium |
| T-Mobile International | Germany |
| Hewlett-Packard Ltd. | United Kingdom |
| Karlstads Universitet | Sweden |
| Università degli studi di Milano | Italy |
| Joint Research Centre | Italy |
| Centre National de la Recherche Scientifique | France |
| Johann Wolfgang Goethe Universität Frankfurt am Main | Germany |
| Chaum LLC | United States of America |
| Rheinisch-Westfälische Technische Hochschule Aachen | Germany |
| Erasmus Universiteit Rotterdam | The Netherlands |
| Stichting Katholieke Universiteit Brabant | The Netherlands |
| Fondazione Centro San Raffaele del Monte Tabor | Italy |
| Swisscom AG | Switzerland |

**Foreword**

# Table of Contents

**1 Introduction** **11**

**2 A High-level View of the PRIME Architecture** **14**
2.1 Roles and Parties . . . 14
2.1.1 Roles . . . 14
2.1.2 Parties . . . 14
2.2 Functionality . . . 15
2.3 PRIME Layer and Application Layer . . . 17
2.4 Objects Held by a Party . . . 17
2.5 Data and Policy Life Cycles . . . 18
2.5.1 Data Life Cycle . . . 18
2.5.2 Policy Life Cycle . . . 19
2.6 Architecture Design Principles . . . 20
2.7 Requirements – The Architectural Foundation . . . 20
2.8 The PRIME Approach . . . 20
2.9 Development Process of the Architecture . . . 21

**3 Trust Model** **22**
3.1 Compromised Applications . . . 22
3.2 Attackers . . . 22
3.3 Limits of Defense . . . 23
3.4 A Weak Trust Model . . . 23
3.5 A Strong Trust Model . . . 24
3.6 Discussion of the Trust Models . . . 24

**4 PRIME Concepts** **24**
4.1 Policies . . . 24
4.2 Data Semantics . . . 25
4.3 Formal-semantics-driven Derivations . . . 25
4.4 Trusted Ontologies . . . 26
4.4.1 Rogue Ontologies . . . 26
4.4.2 Obtaining Trusted Ontologies . . . 26
4.5 Negotiation – How Parties Interact . . . 26

**5 Component Architecture** **27**
5.1 Symmetry . . . 27
5.2 Overview of the Components . . . 27

**6 Data** **30**
6.1 Basic Concepts . . . 31
6.2 Metadata . . . 31
6.3 Implications of Data Exchange . . . 32
6.4 Cryptographic Material . . . 33
6.5 Open Issues . . . 33

**7 Data Model** **34**
7.1 Abstract Model . . . 34
7.2 Relation to Policies . . . 36
7.3 A Concrete Embodiment – RDF . . . 37
7.4 Data Model – Mappings . . . 37
7.4.1 Transparent and Explicit Model . . . 38

**8 Data Exchange** **39**
8.1 Protocols . . . 39
8.1.1 Register . . . 40
8.1.2 Prove . . . 40
8.2 Classes of Protocols for Data Exchange . . . 41
8.2.1 Classical Mechanisms . . . 41
8.2.2 Private-certificate-based Mechanisms . . . 41

**9 Policies** **42**
9.1 Access Control Policies . . . 42
9.1.1 Access Control Policy Language . . . 42
9.1.2 Release Policies . . . 44
9.2 Data Handling Policies . . . 44
9.2.1 Access Control Aspects . . . 44
9.2.2 Privacy Obligation Aspects . . . 45
9.3 Privacy Obligation Policies . . . 45
9.3.1 Making a Policy Enforceable . . . 47
9.4 Assurance Policies . . . 47
9.4.1 Natural Language Examples . . . 47
9.4.2 Formal View . . . 47
9.5 Relation of Policies to the Architecture . . . 48
9.6 The Role of Policies in a Negotiation . . . 48

**10 Negotiation** **49**
10.1 Preliminaries . . . 50
10.2 Protocol . . . 52
10.3 Efficiency and Rationale . . . 55

**11 Assurance Checking** **55**
11.1 Introduction . . . 56
11.1.1 Principles . . . 56
11.1.2 Overview of Different Potential Approaches . . . 56
11.2 Defining Trust Constraints: A Lower-level Representation . . . 57
11.3 Defining Clauses as First Class Objects: A Higher-level Representation . . . 58
11.3.1 Conceptual View . . . 59
11.3.2 Examples of Clauses . . . 60
11.3.3 Operational view . . . 61
11.4 Representation of assurance policies in XML format . . . 61
11.5 Discussion . . . 62
11.6 Next steps and future R&D work . . . 64
11.7 Assurance Control Component . . . 64

**12 Privacy-aware Identity Lifecycle Management: Principles and concepts** **64**
12.1 Introduction . . . 64
12.2 Obligation Management Framework . . . 65
12.3 Obligation Management System . . . 66
12.3.1 Design Rationale . . . 66
12.3.2 System Architecture . . . 66
12.4 Operational View of Privacy Obligations . . . 68
12.5 Discussion and Conclusions . . . 72

**13 Conclusions** **73**

# Table of Contents

## Acknowledgements

The editor wants to thanks every member of the PRIME project who has contributed in one way or the other to the work on the architecture. This addresses everyone who has been involved in technology development in the project, but also colleagues in the legal, social, and economic domains.

Specific thanks goes to Jan Camenisch and andreas Pfitzmann who were always available for technical discussions. Thomas Kriegelstein who has been involved in the architecture and implementation work for several years needs to acknowledged for his effort for the architecture. Marco Casassa Mont and Stefano Crosta need to be mentioned for their work on the previous version of the architecture, particularly an improvement of the componentization and a their perspective on the services-side aspects. Thomas Roessler and Giles Hogben contributed by their experience in the areas of data modeling and ontologies. Other people who are not explicitely mentioned here have made contributions to the architecture.

Many thanks also to Patrik Bichsel for assisting with the production of figures and volunteering for performing a review of the document which was particularly useful due to his fresh perspective on the PRIME Architecture.

## Authors

Sections 9.3 and 12 on privacy obligation policies and system have been authored by Marco Casassa Mont. Sections 9.4 and 11 on assurances has been authored by Siani Pearson. The remaining sections have been authored by the editor.

## Reviewers

As one reviewer we decided to chose Marco Casassa Mont of HP Labs Bristol due to his previous involvement in the architecture process and deep knowledge of many architecture aspects, with a focus on services-side aspects and life-time privacy management. He brought a perspective of an insider into the review process that was valuable in terms of assuring sufficient consistency with the previous version as well as consistency between the user-side and services-side aspects. As this is the final architecture document, we decided to choose also Patrik Bichsel of IBM Zurich who is completely independent from previous work on the PRIME Architecture, but knowledgeable of architecture aspects in general and aware of the foci of and concepts used in PRIME, as reviewer.

# Executive Summary

The PRIME Architecture can be seen as a generic architecture that defines a feasible way of bringing different technologies from the privacy-enhancing technology (PET) space together with the goal of improving the privacy protection for people that interact over an electronic communication network such as the Internet. The architecture goes much further than loosely integrating a set of available privacy technologies, rather our work on the architecture has, during the PRIME project, led to a deeper understanding of how mechanisms work together and also to an improvement of a variety of the involved privacy protection mechanisms.

The added value of PRIME with respect to systems developed by other initiatives can be best summarized as follows: 1) PRIME defines a method for establishing trust by data exchange between two parties in a semi-automated way; 2) PRIME strives to reduce a user's need in excessive trust in other parties such as service providers or certifiers; 3) PRIME allows a user to make a semi-automated assessment of a service provider that allows her to better judge the trustworthiness of the service provider; 4) PRIME provides new approaches to the enforcement of agreed data handling policies with a greater degree of expressivity and automation.

The *establishment of trust* is performed by a mutual release of possibly certified data between two parties. The mutual release of data together with an agreement of data handling policies that need to be applied on those data is called negotiation. The negotiation is to a large extent driven by a new *access control policy* mechanism developed in PRIME. The *data handling policies* of PRIME comprise both aspects that are enforceable by access control and aspects that are unrelated to access control (privacy obligations). The goal of a negotiation is to allow both involved parties to establish sufficient trust in the other party by the provided attribute statements. A wide range of attributes can be used for establishing trust. Examples are certified attributes of persons (as contained in an electronic id card), attributes characterizing organizations (attributes of a privacy seal), attributes characterizing the assurance state of a data processing system (integrity, availability of certain enforcement mechanisms), attributes about the reputation of a party (conduct in previous interactions) and so on.

The *reduction of trust requirements* has been taken up early by PRIME in order to put the user in a better position, thereby reflecting strongly the user-centric approach of PRIME that is directly implemented in the architecture. A key technology for reducing trust requirements is the use of advanced technologies for data exchange. PRIME builds on top of the world's most advanced *anonymous credential system*, identity mixer, that operates in a strong model for identity federation in terms of privacy. Such a system allows a user to authorize with a service without necessarily establishing linkability with other authorizations at this or other services. This reduces the establishment of excessive user profiles by parties that gather user data.

A user can assess various properties of a service provider in order to better judge the trustworthiness of the service provider. The assessment is mainly based on *assurances* that the service provider can communicate to the user on request. Those assurances cover properties of the organization, its processes, and its data protection system in place and its individual platforms. Particularly, the availability of certain protection mechanisms can be asserted to users. *Reputation data* can serve as a useful data source for a trust assessment of another (unkown) party.

The *enforcement of policies* for data once released is taken further in the PRIME project compared with the state of the art. This regards the enforcement of the agreed data handling policies. This enforcement comprises two parts: The more traditional enforcement of access control constraints using advanced policy mechanisms and also the enforcement of privacy obligations by a new approach towards automated enforcement of privacy obligations. The latter has been designed specifically with performance and large-scale data sets and longevity in mind for practical viability.

Overall, we have gained substantial insight into the privacy mechanisms, their interoperability, and related architectural implications during the work on the architecture. We claim to have advanced the state of the art noticeably and to have performed the necessary preparatory work for future deployments of privacy technologies on a larger scale. Particularly, we have achieved a componentization that facilitates a deployment of subsets of the architecture as standalone systems.

# 1 Introduction

Today's information society poses an increasing threat on the people's privacy due to an increasing fraction of interactions being done electronically via communication networks. Personal data are collected in most of the interactions and often stored for extended periods of time and divulged to a set of third parties without people being aware of this. As people leave a data trail in each of their interactions with a single party, an increasing amount of data is aggregated over time at a growing number of parties. Considering this in the light of continually decreasing cost of storage and increasing power of data processing systems, data recipients can establish more and more extensive profiles of people with moderate cost and, from an economic standpoint, keep them for a virtually indefinite time span.

A particular threat to privacy emerges from user profiles held by multiple service providers being merged to obtain even more comprehensive user profiles. This extensive profiling may be undesirable from a user perspective as the users might not even be aware of those data aggregation activities going on. Particularly, we think that too much information is collected in many interactions today and much less information could suffice for many business processes to be carried out. In many interactions, users could even act under a *pseudonym*, that is, not reveal their civil identity, but only certain (certified) attributes of their identity. We refer to this as a partial identity. This can help to prevent, to some degree, the merging of profiles different service providers have from users. For scenarios where a user needs to provide identitfiable data to a service provider, PRIME allows the user to assess the service provider from an assurance perspective and use the assessment as further information to judge the trustworthiness of the service provider.

An issue that is orthogonal to the above-discussed profiling and prevention of the establishment of massive profiles is the protection of user data at the service provider side, that is, the protection of the extensive user profiles once they have been built up. Today's service providers not always have proper mechanisms in place that properly enforce the policy that has been advertised to the user for handling data. For example, violations can occur due to negligence in semi-automated enforcement procedures, e.g., for deleting or anonymizing customer data after an agreed period of time. A completely *policy-driven enforcement* as envisioned in PRIME that helps a service provider to better comply with the advertised data handling policy can support a service provider in mitigating the abovementioned issues of data protection. The policy-driven enforcement is implemented using both access control mechanisms and privacy obligation mechanisms in order to capture the real-world protection requirements for data.

A particular problem in today's e-society is the lack of a unified way of performing identity-related interactions with other parties over electronic communication networks. In other words, each service provider creates their own user experience when it comes to identity management, be it the interface for soliciting attributes, or the way privacy policies are advertised to potential users. One PRIME goal is to propose an architecture that allows for a standardized way of doing user interaction.

**The PRIME project** The PRIME Architecture has been built within the *PRIME project*, a research and development project in the space of privacy-enhancing identity management that is partially funded by the European Commission under the 6th Framework Programme. PRIME stands for *Privacy and Identity Management for Europe*. In the PRIME project we have taken a comprehensive approach towards addressing some of the key issues in the space of privacy-enhancing identity management. PRIME has addressed the problem of how parties can establish trust in each other while at the same time not giving up on privacy. As well, PRIME has addressed the life-cycle management of data, with a particular focus on the phase once users have disclosed their data, that is, the enforcement of the policy under which the data have been released.

**The PRIME Architecture** The PRIME Architecture addresses the above challenges in a comprehensive and integrated way, that is, it addresses an individual privacy issue with the other issues in mind and not in an isolated fashion. This has lead to an architecture that brings together different technologies for enhancing privacy in order to address the problem space broadly and with improved interplay between different privacy technologies.

We give an overview of the key functions of a PRIME system following the PRIME Architecture. Note that we adopt the following terminology: The *PRIME Architecture* is the architecture as presented in this document and a *PRIME system* is an implementation of, typically parts of, the PRIME Architecture. This can be a partial implementation with an appropriate subset of the functionality.

One primary functionality of the PRIME Architecture is to *protect resources*, that is, data and services. For both data and services protection, the basic protection is accomplished by *access protection* to a resource, that is, giving only legitimate parties access to the resource. Access means either consuming a service, or performing an operation (e.g., read, write, or update) on data. This subsumes disclosing data as this is an access by the recipient. For data, the protection goes much further than only protecting access to it: It covers the complete *lify cycle* or data management, particularly the enforcement of the agreed *data handling policy* after data have been disclosed. As a particular example, this includes the automated fulfillment of privacy obligations.

A closely related functionality of the architecture is *negotation* which plays an important role in the access of resources. More concretely, the negotation protocol of PRIME is used on the one hand for *establishing trust* between parties by mutually revealing data and on the other hand to allow for an exchange of required *identity attributes* between the parties. For exchanged attributes, data handling policies are agreed that satisfy the data handling requirements of both parties. The data requirements in the negotiation process are determined by the protection requirements of the resources whose access has triggered a negotiation.

Based on negotiation and resource protection, *access to data* functions are realized by the PRIME Architecture, that is, functionality that allows users to access what data a service provider holds about them and to take corrective actions in case they consider this a privacy violation. This is useful for giving some more control over their data to the users.

**Scope and Non-Scope of the Architecture** We define the scope of the PRIME Architecture and this document as well as the non-scope, that is, considerations that are definitely out of the scope of the architecture. This shall help the reader to get an impression on how far PRIME can go in terms of privacy protection and what is definitely not offered by PRIME.

**Scope**

The architecture for PRIME covered by the current document is the basis for the design and implementation of the *PRIME System* prototype (the *Integrated Prototype*). The prototype comprises a middleware layer for identity management, the *PRIME Middleware*, implementing the various privacy mechanisms adopted and developed by PRIME and the *PRIME Console* implementing the user interface. Both middleware and console together are referred to as *PRIME System*. The Integrated Prototype of the project implements a subset of the PRIME Architecture. Multiple *Application Prototypes* have been built on top of the functionality provided by the Integrated Prototype.

This document communicates the key ideas of the PRIME project in terms of privacy architecture. The architecture takes an integration with legacy technologies into consideration by design at various points in order to facilitate adoption by industry of various of our proposed technologies.

The architecture covers interactions between parties, particularly users and service providers, as far as identity management in an *application-independent* form is concerned. The architecture as well covers generic data processing of user-related data by an organization, regardless of the origin of these data and regardless of whether the data are identifiable. For example, an organization can use PRIME services-side technology such as access control and obligation management functionality that allows one to protect user's data regardless of how the data was entered into the system. This can be used for the protection of data gathered through today's web-forms-based technology on the Internet without the user running a PRIME system and equally well for protection of data gathered via any other means. This gives us additional flexibility in terms of potential services-side-only deployment of PRIME technology. Though, this is not the envisioned form of deployment as the *user centricity* aspect, an important idea in PRIME, is not there any more in such a scenario.

Generally, PRIME applies only to a fragment of interactions of a user with parties in the e-society. This subset of interactions is the set where PRIME technology is used by at least one of the parties. The full potential of PRIME is only available if both parties run PRIME components. The latter case is the case which the architecture document focuses on as well as the preferred deployment scenario.

**Non-scope**

The non-scope of the architecture are interactions happening exclusively on the application layer such as a service provider delivering a service to a user. Such interactions may lead to identity-management-related tasks, however. This is for example the case when users create content online or participate in electronic social networks. Those interactions require adequate handling by the services side and by the respective application in order to not violate the agreed data handling policy.

**Discussion**

We stress again that in case two PRIME-enabled parties interact, only the part of the interaction that is related to privacy and identity management is governed by PRIME. PRIME is only one component of many of a user's computing system or a service provider's IT infrastructure. A service is usually delivered by a business application residing in an application server and consumed through the user's web browser. These software components are not part of PRIME, but use it.

The architecture is definitely not concerned with business processes such as the data processing by services-side applications. The boundary between PRIME and application is where data leave the PRIME system for processing by applications. It is assumed that business applications only do processing according to the data handling policies.

In an ideal situation all service providers for which PRIME technology is applicable would adopt PRIME technology as a standard. Realistically, PRIME technology will diffuse slowly into today's e-business infrastructures of service providers and thus protection will incrementally become available. At the same time, an incremental roll out of user-side technology complements the efforts on the services side.

**Related Work** We refer the reader to the PRIME Architecture V2 deliverable [MCKS06] for a discussion of some related project and standards. Next give an overview of identity management initiatives with a focus on user-centric identity management that have emerged or gained substantial attention within the life time of the PRIME project.

The CardSpace initiative [Mic05] has resulted in an identity management system that allows users to perform the attribute exchange with service providers in a unified way. This is aligned closely with one key goal of PRIME of providing a consistent user experience to people when doing identity-related interactions with service providers. The system can be considered a user-centric system as it puts the user in control of their identities to some extent. The CardSpace client component is contained in the latest Windows platform and thus benefits in terms of deployment. The system is based on the security-related protocols in the WS-* protocol stack. This encompasses particularly *WS-Security*, *WS-Trust*, and *WS-Federation* [KN03].

Another noteable initiative in the identity management space is OpenID [Ope]. The openID project strives for a simple user-centric protocol that allows users to provide identities to web sites. Today's protocol is fully browser based which gives the approach a big advantage in terms of deployment. Though, the user centricity aspects, particularly user control of identity and privacy, are less emphasized than in PRIME, CardSpace, or Higgins. The current protocols suffer from major security and privacy problems.

An group whose main focus is services-centric identity management is the Liberty Alliance [lib] with their identity federation protocols. The Liberty Alliance is focusing on the classical federation aspects, that is, the paradigm where a federation is mostly driven by the involved service providers rather than the users. Though, the Liberty Alliance has also shown its interest in the user-centric paradigm.

A discussion on the different emerging open source initiatives and standards has also been produced within the PrimeLife project [CPR+08]. The main focus of the document lies on the comparison of the techniques and not on the details of the diverse technologies.

**Document Structure** The current document is structured such that the first part up to Sec. 2 allows a reader to get an overview of the basic ideas underlying the architecture. This part is intended also for readers with less technical expertise since discussions are on a rather non-technical level.

Technically-interested readers will find the second part that starts with Sec. 3 useful in getting more detailed insight into the technology underlying PRIME. This technical part discusses details on various architectural aspects, starting with the trust model, various concepts of PRIME, a component architecture, the data representation, data exchange (identity federation), the policy model, the negotiation process of PRIME, particularly covering the establishment of trust between parties, assurances, and life cycle policy enforcement.

We deliberately do not go into implementation-level details in this architecture document as it would increase the document size unproportionally and make it less appealing to many audiences. We refer the reader to Architecture V2 [MCKS06] for an API description.

# 2 A High-level View of the PRIME Architecture

This section gives a high-level overview of the PRIME architecture, including the technical architecture, requirements it is based on, its functionality, underlying design principles, and so on. Particularly, we first discuss the parties and roles in the architecture, the give an overview of the functionality, and distinguish between the PRIME and application layer. We also outline which kinds of 'objects' a party holds, the life cycle of data and policies, architectureal design principles, and the requirements the architecture is built upon.

The PRIME Architecture at a system level is comprised of *parties* that interact. A party may run an instance of the *PRIME System* allowing them to interact with other parties using PRIME protocols and to manage data they hold using PRIME technology. A party can be a user or service provider, the architecture is symmetric for any party.

## 2.1 Roles and Parties

A general architecture for PIM, like the PRIME Architecture, has to deal with multiple different *types of parties* and *roles* that parties can take on in interactions and thereafter. A party may take on different roles in an interaction depending on whom it is interacting with and what is done in the interaction. Multiple roles can be played by a party at the same time. The main gain of using the role concept is that roles can be used for purposes of quick reference to the meaning associated to the role.

### 2.1.1 Roles

We classify roles from some different viewpoints.

From the standpoint of data exchange (identity federation), we have the roles *data provider*[1], *data recipient*[2], *certifier*, (*endorsing party*) and *data subject*. A data provider is characterized by being the party making an identity statement to a data recipient. The data can optionally be certified (endorsed) by a certifier. The data statement thereby describes the data subject. Data exchange goes along with the agreement of a data handling policy between data provider and data recipient to be enforced by the data recipient.

From the viewpoint of who initiates a negotiation, we distinguish the roles of *initiator* and *respondent*. This role persists until the end of the negotiation.

Further specific functionality in the area of data exchange may be available as extensions to the above. One important role is *conditional release trustee*. A party in this role implements functionality related to a specific way of data release, namely providing the clear text of attributes that have been released under a condition to a data recipient. Only in case the condition is fulfilled, the data is provided in clear. This is particularly important in the context of anonymous transactions and allows for de-anonymization of such transactions if required. Thus, a party in this role is required for getting accountability for anonymous transactions. The following gives a rough overview of different types of parties and what their stakes in the PRIME Architecture are. From the point of view of service provisioning, the roles of *service provider* and *service requester* with the canonical meaning can be distinguished.

### 2.1.2 Parties

Orthogonally to the role concept, we classify parties statically. We consider two main classes of parties in PRIME: *users* and *organizations* (*service providers*). This distinction is made mainly to allow discussing the data protection requirements and law-defined obligations for individual classes of players and to illustrate scenarios and examples. The distinction is not technically or architecturally motivated as the architecture is largely symmetric. From an architecture point of view, each player is represented as a party, that is, each player potentially has the same functionality in terms of privacy protection of data. Though, in a real world implementation, implementations of users and organizations will greatly differ, regardless of their conceptual similarity. This is due to requirements for scalability and enterprise integration on the service provider side. Users take on the role of service consumer, organizations the role of service provider in most of the interactions with each other and in all examples in this architecture document.

[1] The data provider is often called *requestor*.
[2] Note that a data recipient is often called *relying party* in other contexts.

**User.** A user is a natural person (end user) who is protected by various data protection laws such as the European Data Protection Directive and its implementations in the EU member countries together with her data processing devices, both of them residing in a specific environment at a particular point in time which defines contextual information for privacy management (e.g., time or location).

A user performs interactions with other users and organizations. A part of the interactions are related to identity management and thus performed by the user's PRIME middleware in a privacy-enhancing manner. A user's PRIME system provides the user a PRIME Console for interactions with the system. This Console is a privileged application interacting with the middleware allowing the end user to control the PRIME middleware.

In a *user-to-organization* scenario the user acts as a service requester and the organization as service provider. These roles are fixed throughout the action.

In a *user-to-user* interaction, either involved party acts as a peer, that is, as a service requester and service provider within the same interaction. In such scenarios it may be possible that both entities request services from each other and release data with agreed policy to each other, and apply enforcement mechanisms on each other's data. In a user-to-user interaction, no user has particular legal obligations following from today's legal framework in the EU. Nevertheless, the policy-driven enforcement mechanisms and trust establishment mechanisms of PRIME are applicable to such scenarios and help increase the deployment of peer-to-peer services as peers can get higher assurance than today that their data will be handled appropriately.

For users PRIME offers new ways of authorizing for accessing services provided by other entities which can either be other users or organizations. The new mechanisms allow us to shift from the paradigm of using identifying user attributes to the paradigm of authorization through (certified) partial identities. Partial identities can be identifying the users in certain cases where this is required by the underlying business process. Users can obtain convincing assurances from an organization that they will enforce the data handling policy for the purpose-bound use of the released data agreed with the user. This is particularly important in the case of releasing identifying data. Another key aspect is that different identifiers (pseudonyms) be used with different entities the user interacts with and associated with partial identities.

**Organization or service provider.** An organization (org) is a legal person and their data processing machines providing services to users and other organizations. PRIME middleware can run on multiple of the machines. For providing services to users, an organization typically has to solicit some kind of data from the user. Within Europe, these data are subject to the European Data Protection Directive and various other laws.

PRIME offers organizations new mechanisms of authorizing users by requesting exactly the attribute information about the user that is required for the authorization decision to be made. It is in many cases not necessary—in contrast to today's practice—to solicit attributes that make the user identifiable. This reduces the risks of privacy breaches as there are no person-related data there to be leaked and thus reduces the risk of prosecution through law enforcement and reputation damage. In the case that data that are to be protected has to be released for a particular business process, the PRIME middleware provides functionality for agreeing on a data handling policy with the user and subsequently enforcing this data handing policy locally against other potential recipients of those data. This reduces the risks of inadvertend disclosure to unintended recipients inside and outside the organization. In addition, the user-agreed purposes of the data are enforced.

In an *org-to-org* interaction, data about users can be conveyed from one org to another for facilitating the business process. PRIME provides policy mechanisms to help orgs to remain compliant with the user-agreed policies.

PRIME comes in the form of a middleware component and user interface component that implement the PRIME functionality. Figure 1 gives a high-level architecture diagram of the system of a party, including applications and the PRIME components.

## 2.2 Functionality

The PRIME Architecture is designed to support a comprehensive life-cycle management of identity-related data. A particular focus is on supporting users to *manage* their identity data, in particular their (partial) identities. From a life-cycle perspective, management of data encompasses their creation, their

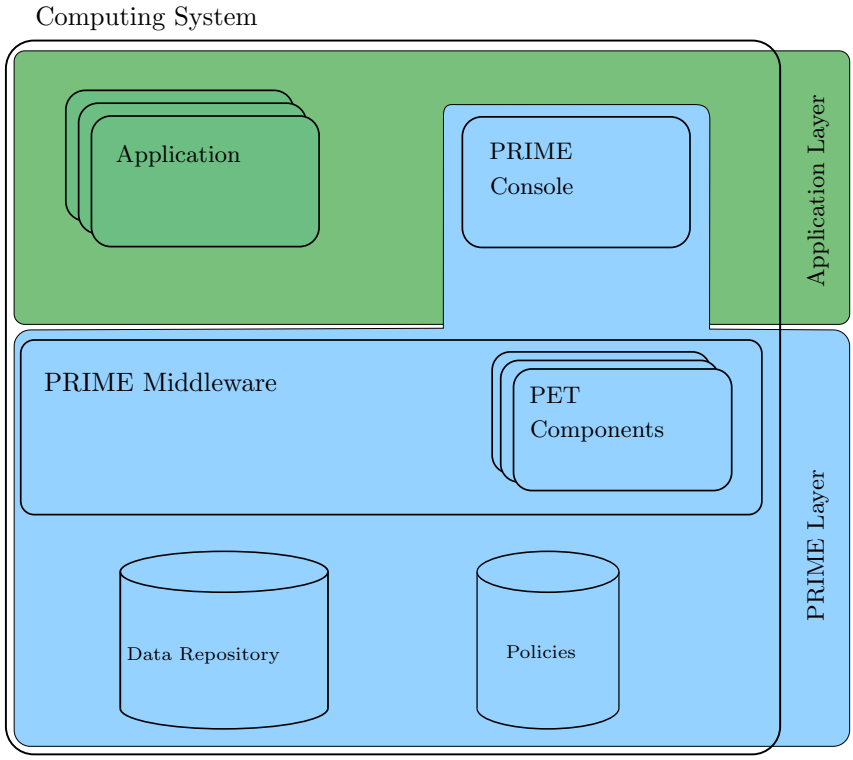


Figure 1: Software Architecture of a Party

use, and their disposal, at each concerned party. We strive to support full coverage of the identity data life-cycle to fulfill requirements from both user and business process perspectives.

Parties are supported by PRIME technology in the management of their own and other parties' data that they hold. This includes access to the data and release to other parties as key functionalities. The operation of the system is policy driven to a large extent, and can in addition involve humans in order to account for aspects that are not modelled in policies.

We next provide a non-exclusive list of the functionality of a party, either for interactions with other parties or for local privacy management:

- Connections
  - Establishing a connection to another entity. Connections are by default done via an anonymous communication channel to avoid leakage of information. Connections can be established between applications and between PRIME systems.
  - Termination of a connection with another entity.
- Resource (services and data) request and release
  - Requesting a resource from another entity. This includes the request of a service or of data statements. A service can be a service provided by applications or an identity management service provided by PRIME.
  - Releasing of data statements to another party. In particular, this includes certified data based on private certificates (anonymous credentials).
  - Accessing one's data that is held by another party. This includes requesting rectification, blocking, and deletion. This is modelled as a special service one the PRIME middleware layer.
- Request and issuance of security tokens like private certificates. The issuance can be an interactive protocol.
  - Requesting a security token.

  - Issuance of a security token.

- Policy-related activities
  - Agreeing on a data handling policy for data to be released. This takes the data handling preferences of the potential data provider and the data handling policy of the potential data recipient into account.
  - Computing the authorization decision for an access to a resource (service or data).
  - Enforcement of agreed data handling policies. This include the enforcement of the authorization rules for accesses to data and the enforcement of privacy obligations.
  - Enforcement of the party's authorization policies on accesses to data.

- Local data handling
  - Storing data received from a party in association with the agreed data handling policy.
  - Deciding on how to fulfill a request for data.
  - Retrieving locally held data.

- Executing a negotiation protocol with another party.

## 2.3 PRIME Layer and Application Layer

As a basic property of PRIME, a PRIME system cannot manage identity data that is communicated on the application level independently of PRIME. This is important to note as in many application scenarios a substantial amount of identity data, possibly even identifying data, is communicated at the application level after an initial trust establishment using PRIME technology. These data can contain user attribute data such as in electronic social networks, or other content such as in chat software, blogs, any other kind of content creation. The communication of such data can violate anonymity and needs to be kept in mind.

Parties interact with each other by exchanging messages. A part of those interactions is handled by the PRIME systems of the parties, another part by the applications of the parties. For example, when a user logs on to a forum, the logon is handled by PRIME performing a data release interaction, and when the user than posts entries on the forum, this is handled on the application level. This separation is motivated by carving out identity-management-related parts of the interactions and moving them to the PRIME level. There is an application-specific border line to be drawn here which is not a simple task as many applications need to handle identity data as their inherent functionality (e.g., electronic social networks). For such applications it is typically not feasible to handle all identity release actions on the PRIME level as this would impair the usability of the application as the application has very special user interaction metaphors to provide a streamlined user experience. PRIME, on the other hand, implements a trust establishment protocol as one of its key functionalities which is not applicable to those application-layer identity management actions.

Interactions on the PRIME and application layers can take part interchangeably during one session. E.g., a new data release comprising different attributes after some application-level interactions can be required to gradually reveal more information to the communication partner.

## 2.4 Objects Held by a Party

Each party holds 'objects' (in a conceptual sense) of a number of different types and uses them in performing its interactions with other entities and its local data management tasks. We elaborate on what is needed in a comprehensive identity management architecture like the PRIME Architecture.

- Data and metadata: Those are considered to be resources where policies are attached to them in a fine-grained way. Details are described in the section on data modeling (7). Data and metadata are treated the same as this allows for advanced identity management functionality such as making the enforcement metadata for user data available to this user only. Data can pertain to the party itself or any other party.

- Access control policies: The policies apply to the resources (services and particularly for the data). The policies govern any kind of access, including data release, to resources. Considering the way the policies are used in interactions with other parties, the name negotiation policies would also apply well. Considering the use of negotiation for trust establishment, the name trust policies can be motivated, at least for those parts of the policies that specify requirements of potential service or data requesters.
- Assurance policies: This kind of policies specify either *assurance statements* of what a party supports, or *assurance requirements* of a party in terms of which assurances it requires from another party. Assurance statements can, more abstractly, be modeled as data statements in the calculus PRIME uses, and assurance requirements can be modeled using access control policies with an extended set of predicates to express the specific assurance-related requirements. We give separate mention to assurance policies due to their importance in PRIME. Also, for technical reasons of the involved protocols, they are handled in a dedicated component.
- Data handling policies: The data handling policies can be defined on ontology types, any abstractions in an ontology, or instances of data. Data handling policies comprise rules covering two different aspects: *Access control aspects* that are enforced by the recipients' access control system and *privacy obligations* that are enforced by an Obligation Management System.
- Activated privacy obligations: The data handling policies only specify privacy obligations in terms of their requirements and agreement. The privacy obligation part of data handling policies is not enforceable itself. Once data have been released with associated data handling policies, the privacy obligations need to be 'pushed' to the Obligation Management System in order to become active in the sense of becoming automatically enforceable by the Obligation Management System.
- Security tokens: Security tokens are used to facilitate interactions between parties and for securing data locally. For example, a security token can be used to prove that a third party endorses a statement and thus to inject more trust in the data. Examples are certificates, cryptographic keys, cryptographic proof tokens, traditional federation tokens. A data encryption key contained in a secure key store and used for encryption of data at the party is an example for a security token used locally.
- Ontologies. The ontologies of the party are used to make derivations on data over those ontologies, e.g., within access control decisions or data release decisions. Ontologies need to be trustworthy as changing them maliciously could be as detrimental to privacy as introducing malicious code. Ontologies can be applied to data of any category, including user attribute data, assurance data, and reputation data.

Note that in addition to the above, parties can provide services to other parties access to which is governed by the PRIME Architecture and access control policies defined over the services. PRIME can only provide the decision for an access to a service, the enforcement of the decision is left to the application layer.

## 2.5 Data and Policy Life Cycles

Throughout the PRIME Architecture, data and policies are concepts that appear in many contexts. Thus, we present a simple lifecycle for data and policies and map them to the PRIME Architecture.

### 2.5.1 Data Life Cycle

We briefly present the notion of data lifecyle and map its phases onto the high-level PRIME Architecture.

**Creation.** The lifecycle of data starts when the data is created, i.e., enters the system. This can be done within an entity by inputting the data into the data processing system. An example for this is a user who enters attributes of themselves into their PRIME Console to have them stored by PRIME. An example on the organization side is the semi-automated reading of non-electronically stored customer records into the data processing system of the entity. An example for implicit creation of data is a user creating a clickstream history on an organization's server by their browsing. This history is stored by the organization's application in their PRIME system.

**Use.** The use of data captures any processing of the data, including release to other entities. Processing can be reading, changing, or utilization of data for a business process such as shipping products. Whenever data regarding other entities are used, this must happen only within the purposes in the agreed data handling policy.

We can distinguish between *local use* and *release*. Local use is the use of data within an entity, release if the transfer of data to another entity. That is, through a release, data crosses the boundaries between entities. Through the release of data, typically a new instance of the data is created at another entity. A special case of release is when an entity accesses their data at a remote entity.

Considering release, there are three main ways of entity-related data of an entity A being gathered by an entity B:

- Explicit release by entity A (e.g., by filling in a form or via the PRIME-defined trust negotiation process);
- Implicit disclosure by entity A (e.g., by clicking through a web site's content, clickstream data is released; IP address of A if A is not using an anonymizer);
- Data regarding A received by B from a third party C without involvement of A.

The data regarding a user is also called instance data further on in this document. Data is instance data regardless of the nymity of the user, i.e., instance data can refer to a user that is known only to a pseudonym. The PRIME middleware supports any of these ways of gathering information and provides mechanisms for a user to decide what information to disclose.

Explicit disclosure is ideally only performed within PRIME-controlled interactions between parties. This allows to seamlessly apply the protection mechanisms to the data. Implicitly disclosed data are gathered by the service provider and stored according to the PRIME data model. The protection mechanisms of PRIME are equally applied to these data. Data that is received from third parties is handled analogously.

Sometimes we refer to instance data as Personally Identifiable Information (PII), although it is not identifiable right away. We note that the Data Protection Directive applies to identifiable data only, but not to data that is / has been made anonymous or pseudonymous.

**Deletion.** The deletion is the final phase of the data lifecycle. Data should be deleted if they are not required for any of the purposes they have been stored for any more. Deletion of data regards one instance of the data only. Deletion of data is typically governed by the privacy obligation part of a data handling policy.

### 2.5.2 Policy Life Cycle

Similarly to the data lifecycle, we define a policy lifecycle and map it to the high-level architecture.

**Creation.** The creation of a policy happens at an entity, e.g., by a human entering the policy. An entity can create their own policies or rely on other entities to create their policies. For example, many users will rely on the vendors of PRIME software to ship predefined policies, or get their policies from consumer protection agencies. An org typically creates their own policies as required for their business process and to comply with laws.

**Use.** The use of policies makes a distinction between the use for local evaluations of the policy and the communication of (parts of) the policy to other entities. The first case is the local enforcement of authorization or data handling policies to govern accesses. The second case is the negotiation of a data handling policy between the involved entities where each of the entities has to reveal partial information on their policy. The trust negotiation process covers both of the cases. Communicating a policy to other entities creates new instances of the policy at those entities.

**Deletion.** The deletion of a policy deletes the current instance of the policy.

## 2.6 Architecture Design Principles

We present the key underlying principles that have guided the development of the PRIME Architecture. This motivates the comprehensive approach taken by PRIME in terms of data protection.

The design of the architecture follows the basic rule of keeping the architecture *open* for future extensions. This goes hand in hand with the approach of keeping the architecture *modularized* and keeping a good functional separation between the major components. This leads to an overall design that is flexible such that is can be adopted to upcoming technologies or changes within the technologies employed. Another guiding principle is *simplicity*. The latter is particularly important for getting outside people interested in using it. The overall design is done such that interactions of users with other parties start with maximum privacy and data is revealed only as necessary. This is a general principle of the PRIME project and is relevant throughout our work.

## 2.7 Requirements – The Architectural Foundation

The driving force of the architectural work was to a large extent requirements imposed by law or elaborated in other areas of the PRIME project. These requirements have acted in various iterations as driver for the development of the architecture. The requirements work in PRIME has considered common use cases, or classes of use cases, often represented by an individual use case. Via this approach, the areas where privacy problems are prevalent were identified and then prioritized to become input to the architecture work.

The European Directives 95/46/EC [Eur95], 2002/58/EC [Eur02], and the data protection laws of the European member states that implement these directives form the basic legal foundation of the work on the architecture. The most project-relevant parts of these and other relevant directives and laws are summarized by the PRIME deliverable on legal requirements [Buc04]. The PRIME Framework in its different releases [FHA04, FHAH05, LFH06, FHH08] provided a useful conceptual frame around our work.

Requirements in the field of target application scenarios for PRIME have been put forth in the PRIME deliverables on application requirements [Wil04, Buc05] and the final requirements [Sch08]. The application requirements describe scenarios in different fields such as e-commerce, location-based services, airport security, healthcare, and e-learning. A subset of these classes of scenarios has been chosen for being implemented as application prototypes.

## 2.8 The PRIME Approach

Based on the issues mentioned in the beginning and the requirements coming from other activities within the PRIME project, we stress the need for a privacy-enhancing identity management system a la PRIME and next give a non-exhaustive overview of what functionalities a privacy-enhancing identity management solution like PRIME needs to provide.

In short, such an identity management solution should give users much more control over their data and help them to make well-informed decisions on the disclosure of data. Also, user-defined policies should support the users in their interactions and take off a major part of the identity management work. In particular, the users should be given a tool to allow them to even maintain control over their data after the release of data by agreeing on a policy for the data handling with the data recipient and by getting confidence in the recipient in enforcing this policy. Furthermore, the system should allow users to access their data after disclosure, e.g., for rectifying, blocking, or requesing a deletion.

- The user-side system by default uses an anonymizer in order not to reveal the user's network address. This can be turned off if desired by the user, e.g., for bandwidth reasons and for the reason of other implications of the trust model of an anonymizing overlay network such as the one by the TOR project.
- Data handling policies are expressed in a formal notation with well-defined semantics.
- Data handling policies for a specific interaction are communicated to the user side and analyzed automatically there. It can be subject to negotiation between the user and services side.
- A standard user interface is used by the user for all privacy-relevant activities. This interface is part of the user's PRIME software and thus a trustworthy interface under control of the user and it

provides a consistent user experience throughout all interactions with PRIME-enabled entities. The idea is to have all privacy-related functionality be handled by this interface and not by applications. Today's web-based interfaces are a counterexample to this approach of a trusted interface under the user's control.

- The data handling policy is presented to the user in a standard way using the PRIME user interface of the PRIME Console. The user-targeted representation of the policy is automatically derived from the formal representation. The transformed policy is in an easily readable and understandable form. Thus the user need not read complicated policies. Overall, this leads to a more well-informed decision on whether to consent to a particular release of data under a certain policy.

- The user side can automatically assess assurance properties and better judge the trustworthiness of the service-provider-side system. The trustworthiness of a party's own system can be assessed as well, for example to determine whether it is an uncompromised trusted computing platform.

- The user can reveal certified data in a data minimizaing way. That is, the user can proof that certified attributes are associated with her without revealing anything else besides these attributes. This can be done without the certifier learning anything about the interaction taking place and the interaction is unlinkable to other interactions the user makes unless attribute information allows for establishing linkability. This can replace the disclosure of identifying information in many scenarios if such certified attributes are enough for a business process to work. It allows the user to often remain pseudonymous or anonymous in such transactions. Overall, this approach goes beyond what currently deployed technologies can do for achieving a data exchange system (federated identity management system) with much stronger privacy properties.

- Enforcement mechanisms on the services side are available that ease services-side data management. Examples are access control mechanisms that enforce the privacy policy subject to which the data have been disclosed by the user when user data is being accessed on the services side. Privacy obligation management allows to automatically enforce the privacy policy aspects that are orthogonal to access of data (conditions), e.g., by the deletion of data after an agreed period of time. The enforcement mechanisms make it easier for an organization to be compliant with agreed policies without the otherwise inherent risks of unintentional policy violation.

- Many of the steps required in an interaction between a user and service provider are automated on the user side, including rule-based decisions regarding the disclosure of data and negotiation of policies. The user is still involved if this is required by law (e.g., for giving (express) consent to data processing) or by the user's preferences. This semi-automated approach allows for not bothering the user with tasks that can be automated, but still involving her into decisions whenever necessary.

- The user manages all identity-management-related data in a centralized data store where any access to it is goverened by the PRIME access control subsystem. The data include the interaction history which allows to keep track of all data disclosures and retain an overview of the data disclosures over time.

Considering the abovementioned features of a PRIME-based identity management solution, such a solution exceeds current practice on the Internet and Web by far in terms of privacy support. The user can, using such a system, make well-informed decisions based on precisely stated and automatically processed policies. The user's decision can be supported by the PRIME software's policy-based decisions.

## 2.9 Development Process of the Architecture

The PRIME Architecture has gone through a number of iterations in the course of the project. The initial release *PRIME-Arch-V0* was mainly targeted at a project-internal audience for the reason of tight limitations of production time. *PRIME-Arch-V1* was intended to be a consolidated version with better coverage of all the key aspects of identity management that are subject to the work within the PRIME project. The *PRIME-Arch-V2* release was targeted more strongly towards an external audience by giving less implementation-related details, and covering concepts on a broader basis. The final *PRIME Architecture V3* as represented by this document continues the approach of being targeted more strongly towards an external audience. This release focuses on providing an overview of the architectural results we

have obtained in the project while trading off the detail level and document size to address our intended audiences in a best possible manner. Note that the current release V3 does not make obsolete the release V2, but rather leaves V2 to present the reference architecture and some implementation-level details, such as APIs, for the PRIME Architecture. In addition to this, V3 provides deeper insight into various concepts and technologies and elaborates on the dependencies of those while still keeping the discussion at a not too specific level. The overall goal of V3 is a consolidated view of the most important aspects of an architecture for a privacy-enhancing identity management system.

Note that specific details of the architectural work that has been performed within the PRIME project are not elaborated on in this document, but reflected in publications published within the scope of the project. The architecture document series presents the umbrella around all the work done with an attempt of summarizing our efforts it in a form that is digestable by our audience and helps the project in conveying the ideas of privacy-enhancing identity management to the interested readers.

# 3 Trust Model

This section talks about the trust model PRIME is based on. This is important information in terms of it being the baseline security and trust are assessed against. One key point is that parties trust their PRIME system. That is, a party can assume that their local PRIME system operates as advertised and does not compromise privacy in an unintended way. We do not give a formal trust model here, but rather focus on givin the reader the intuition of it as this is the most facilitating approach for our intended readership.

We present a discussion on how much protection against compromised applications we can offer in PRIME, a general discussion of attackers in a PRIME setting, followed by a weak and a strong trust model and their relations to the PRIME Architecture.

## 3.1 Compromised Applications

It depends a lot on the underlying system of whether and to what extent rogue applications can be acceptable. An application on a user's machine can only access the PRIME system using the API of the PRIME Middleware, thus a user would notice certain actions that the rogue application would trigger in PRIME components due to the PRIME Console being shown to the user. At a servive provider, a compromised application can achieve much more as the PRIME system operates without user intervention and a rogue application could freely access the interface of the PRIME subsystem unless those actions of misuse would raise an alarm, e.g., raised by a runtime monitoring system.

## 3.2 Attackers

A complex environment as given in today's e-society allows for plenty of opportunities for attackers. Generally, we consider as attacker or adversary a party who may obtain more information on parties (particularly users) as they should. This notion holds regardless of whether the attacker operates on legal grounds – this is an orthogonal issue.

That means attackers can come in many flavours. For example, an attacker can compromise a single point in the system such as a user's computer system by means of a trojan horse, virus, or hacker attack in order to obtain data about the user of the system. On the other hand, an attacker can be a coercion of multiple service providers and certifiers who try to build up comprehensive profiles of users they are having interactions with by pooling user profiles. Content aggregators are a particular example for such an attacker, operating on legal ground. Another type of attacker is represented by network providers who can build up massive profiles on their users' interactions through traffic monitoring. Targets in terms of data are personal attribute data, private keys, access credentials such as passwords, connection profiles, and other user-related information that can be used for (illegitimate) purposes.

As a general classification, one can distinguish passive adversaries and active adversaries. A passive adversary always follows its protocols with other parties as defined, but tries to extract as much information from the information it legitimately gains in its interactions. For example, passively dishonest service providers are such that share their interaction transcripts within coalitions of service providers and can infer more information on users by aggregating the shared profiles than each one could do on their own. Another example for passively dishonest parties are service providers who do not delete user

data as agreed with the user. Passive adversaries are known as *honest, but curious* parties. An active adversary can deviate from its specified protocols in interactions with other parties in an arbitrary way. This includes deviating from (cryptographic) protocols executed with other parties in an arbitrary way, probably to gain an advantage.

## 3.3 Limits of Defense

The technology we have developed and the PRIME Architecture are targeted towards mitigating many of the issues arising by both passive and active adversaries. One key defense is the use of unlinkable transactions. Unlinkable transactions carried out by one user with one or multiple other parties have the property that the other parties cannot determine, for any of the transactions, whether they have been carried out with the same user. The key property of PRIME in this space is that if some kind of authentication with the other parties is required, this authentication can be accomplished on the basis of certified attributes without establishing linkability, of course only unless the attribute themselves establish linkability. The idea is to provide exactly the attribute information required by the other party in a certified form. The unlinkability holds even against active adversaries in PRIME.

It is hard to defend against malicious parties in the general case, e.g., if a business interaction requires personal data to be revealed to an adversarial party and if a data handling policy is to be enforced by this party. This enforcement cannot be guaranteed with current technology, but integrity-protected computing[3] might help in the future. The use of integrity-protected computing may be specifically useful in making attacks by insiders on the service provider side more difficult. Though, a malicious party is always capable of circumventing controls in some way and misuse personal data released to them if they do an appropriate effort.

For a large fraction of service providers one can assume that they are honest and willing to fulfill the agreed policy, that is, to act as specified. Though, mistakes happen in the policym enforcement process and technology should be there to prevent those. This means, defending against unintentional mistakes that may happen due to negligence or inappropriate enforcement systems, can greatly improve compliance with advertised practices and raise the barrier for attacks, e.g., by insiders of the company. Two examples from the PRIME Architecture that improve on policy compliance are the *privacy obligation management* and the *access control* systems.

## 3.4 A Weak Trust Model

We now present the ideas of a minimal trust model, that is, the weakest one that can still be considered useful for general-purpose identity management. We cover three key aspects: Communication, data exchange (identity federation), and enforcement of the data handling policy.

In terms of *communication* we assume that parties are eavesdropping on the network, that is, a need for a secure channel arises. This can be resolved using standard technologies in order to prevent an eavesdropper to learn data they should not learn.

There are some basic requirements for a *data exchange* protocol that need to be fulfilled in every identity federation system: First, the integrity of certified identity information is preserved, that is, the statements that are endorsed by a certifier cannot be changed by any party without this being detectable. Second, the exchanged data must remain confidential to the involved parties in the data exchange process, namely the data provider, the data recipient, the certifier, and the data subject. Third, data recipients need to rely on the certifier to provide correct attribute statements. The latter is the underlying assumption that makes federation systems work at all. See Sec. 8 for details on data exchange protocols.

The idea of enforcement of *data handling policies* is, considering today's computing architectures, based on trust in proper behaviour of the party that is supposed to enforce the policies. That is, a dishonest party can arbitrarily violate the agreed data handling policy with legal enforcement being the only after-the-fact defense against this.

The weak trust model sketched above represents the one on which today's electronic interactions rely on in the best case. In the view of PRIME, too much trust needs to be put in parties in this model, e.g., trust in proper enforcement of the agreed policies. We think that we can reduce the trust requirements in parties while not compromising on functionality.

[3]We use the term *integrity-protected computing* for what is known as *trusted computing* as it reflects better what can be offered by this technology and does not use the term trust with its many quite different meanings.

### 3.5 A Strong Trust Model

Next, we present a strong trust model that reflects the vision of PRIME of reducing trust requirements in parties such as service providers and certifiers. PRIME provides for functionality to satisfy this model under certain circumstances. Overall, this model assumes adversarial behaviour of all parties.[4] Particularly, an attacker can be formed of an arbitrary coalition of the involved parties.

We note that no system can be built with current technology that is secure in such a trust model in all use cases. To be more particular, whenever identifying personal data need to be released under such a trust model, the data are to some extent subject to misuse, particularly user profiling and unlimited sharing of those data. Though, this problem is mitigated to some extent as in many interactions it is sufficient to release non-identifying data in certified form.

**Example** For example, the fact that a user has purchased an electronic subscription to the online resources of a scientific publisher is sufficient for the publisher to decide on whether to grant an access request to online content. In this example scenario, PRIME can provide the capability of a user proving ownership of a service subscription without being linkable or identifiable. Anonymous credential systems make this possible. This even holds in case of the service provider being arbitrarily dishonest in its behaviour. In this case there is no risk of data misuse or profiling as the data cannot be assigned to the user due to the lack of identifiability. In more general setting, this holds even if all service providers and certifiers behave dishonestly and collaborate in achieving their goals of creating user profiles.

### 3.6 Discussion of the Trust Models

To conclude the discussions on trust models for PRIME, it should be noted that no single trust model is sufficient for capturing the needs of all relevant scenarios yet being implementable by any real-world system. For this reason, different trust model should be applied in different scenarios, e.g., depending on whether identifiable data are involved. A conservative approach is to operate in each single interaction in the strongest trust model for which a system exists that is secure under this trust model.

## 4 PRIME Concepts

We elaborate on the different concepts that are underlying the PRIME Architecture and have influenced its construction. The goal of this section is to familiarize the reader with the foundational concepts and ideas we use in PRIME.

### 4.1 Policies

Data management and interactions in PRIME are governed to a large extent by policies. We next give an overview of the different types of policies we have used in the PRIME Architecture.

*Access control policies* govern the access to resources of a party, where resources can be data or services managed by the party. An access control policy rule applies to a set of objects and specifies which subjects may perform which operations on the object under which conditions.

*Release policies* are similar to access control policies, but define the circumstances when data may be accessed to release them to another entity.

*Data handling policies* govern how data must be handled by the recipient of data. Data handling policies may be imposed on data by the data subject and further DHPs may be added by the data recipient to account for their data management practices. When data are released by a recipient to third parties, the original DHP may only be specialized. A specialized policy enforces the original policy, that is, it is at least as strict as the original one.

[4]Clearly, this does not mean, that we assume that all parties in practice do in fact behave adversarially. Just in case certain parties do, a system that is secure in this trust model still provides for appropriate protection of a user's data in this case. Overall, this means that more open system can be built when assuming this trust model as also parties without a priori reputation can be safely interacted with.

## 4.2 Data Semantics

As identity management deals with data of entities, particularly users' data, data modeling is of prime importance for the architecture. The data model defines how data are represented on a conceptual level and gives semantics to the data.

The data model gives commonly understood semantics to the data; this is one of the key features, particularly from an interoperability point of view. This common semantics on data is particularly important for an interoperability point of view, that is, for exchanging data. Not less important, this semantics specifies how policies are associated to data and how policies may apply to data. Thus, the data model is relevant as well once an interaction has been successfully performed and data are managed locally by a party. Also, further disclosures of data, e.g., by a service provider to another service provider need the semantics.

The data model does not mandate a specific implementation, unless data are being transferred between parties and interoperability is required. The latter requires that a standardized approach towards how data are modeled in inter-party interactions. In PRIME we use the same data model for representing data within a single party and in operations that span over two or more parties. In practical system deployments, large service providers that are handling vast amounts of user data may use models that are specifically designed for better efficiency when handling local operations.

Some basic properties of the data model are the following ones, based on requirements from an identity management perspective. We assume the availability of a system for providing names to types and resources without namesapce clashes. W3C's Uniform Resource Identifiers (URIs) are a suitable candidate for such a system as they are based on a simple yet effective and widely-deployed mechanism for namespace management. Data are conceptually structured in a way that allows to refer to full records (profiles) or parts thereof or single attributes or data units. This is required for associating policies effectively with data. Furhermore, attribute data can be enriched with *metadata* that makes statements about the data, e.g., who endorsed a statement.

## 4.3 Formal-semantics-driven Derivations

PRIME uses multiple types of *ontologies* to formally express the semantics of concepts and their relations. An ontology is, abstractly speaking, a representation of concepts and their relations. A concept can be a data category, such as 'address data' and a relation can be a statement like 'address data are personal data'. Another example for a relation is that the purpose 'applied research' is a sub-purpose of 'research' which can be useful in the definition of policies. Multiple ontologies are defined within the PRIME Architecture in order to account for different needs in the data management life-cycle.

*Data ontology*: (also called *data type ontology*) Specifies the data types and their relations in PRIME, including data abstractions such as hierarchies between types and categories and among categories. A core data ontology is specified and commonly used by all players as a 'built-in' default in order to ensure smooth interoperation as far as the most commonly-used data types are concerned.[5] Further data ontologies can and need to be obtained in case they are needed. This allows for an open identity management system.

*Object ontology*: A service provider has their own object ontology in order to structure the resource space of their objects that can be accessed by requestors. Object ontologies help express the policies more concisely and are typically not shared by different parties. See Bonatti and Samarati [BS02] for details and an example for object ontologies.

*Certification ontology*: A certification ontology expresses the properties of certifiers (identity providers) in an abstract way. This helps express policies in a more abstract way than would be possible without the use of such an ontology. A good example for what is modelled in a certification ontology is the following informally stated one in the scope of government-issued identity cards: 'A German id card is a EU-Government-issued id card'. Having a certification ontology talking about all different kinds of EU id cards makes identity federations more open and policies easier to express with respect to talking about the trust in attributes. More concretely, by using such an ontology, data requirements in policies can be phrased in terms of abstract concepts (EU-Government-issued id card) and still using a more concret concept (German-Government-issued id card) in an interaction fulfills the abstract one.

[5]Note that the CardSpace specification features an exhaustive list of less than 20 ontology types for their system while not providing any ontology to structure them.

There are multiple issues to be discussed on more detail in the context of ontologies, regardless of what ontologies we talk about, in particular mechanisms how multiple players agree on ontologies to use in addition to the basic PRIME ontologies. See Sec. 4.4 for a discussion of some challenges in an open ontology system.

## 4.4 Trusted Ontologies

When using ontologies for formal-semantics-driven derivations as explained, it is important to have an architecture in place that allows parties to obtain ontologies they need for those purposes. Particularly, a requirement is that no rogue ontology is accepted as this would compromise the security of the overall system. This section is intended to convey the basic challenges, but we will not go into technical details.

### 4.4.1 Rogue Ontologies

A *rogue* ontology is an ontology that is crafted maliciously by an adversary who wants to attack the system to achieve a certain benefit, e.g., access a resource without being eligible or make another party reveal data that they do not intend to reveal. The goal of the attacker is to let a party accept a rogue ontology as a trusted one such that the party will base a subset of its decisions on this ontology.

Accepting a rogue ontology is comparable to executing rogue code on ones machine as it changes the execution behaviour of the identity management program. This is the case as a multitude of security-critical decisions are based on ontologies and thus behaviour is determined by the ontologies. This makes the importance of a clean architecture for securely handling ontologies explicit.

### 4.4.2 Obtaining Trusted Ontologies

For the abovementioned problems that come with rogue ontologies, the system must provide means for obtaining an ontology in a way such that the ontology can be trusted by the party for the purpose at hand.

A simple means of obtaining an ontology with all its advantages and disadvantages is by means of using standard PKI technology. Then a user or a server can, for example, obtain their ontology by their government who acts as an ontology provider. The problem of using just such a simple approach is that it is not sufficiently open and suffers the problem of PKI. A particular problem is that ontology providers are classified in trusted and untrusted. This is not sufficient for an open system.

A more sophisticated and open way of handling the secure distribution of ontologies is to leverage the idea of using ontologies for making decisions on the trustworthy distribution of ontologies. Basically, our method includes a new type of ontology denoted *Provider Ontology* that makes statements on the trust in Ontology Providers. There are different approaches of how Provider Ontologies can be used in order to secure the distribution of ontologies while at the same time allowing for openness of the system. See [HS06] for a discussion of ideas that PRIME has elaborated on.

## 4.5 Negotiation – How Parties Interact

Interactions as sketched in this subsection are performed whenever two parties start a communication, be they known or unknown to each other at that time. The parties execute a so called *negotiation* that allows them to assess the other party's trustworthiness for the interaction and to acquire attributes needed to execute their business process. Note that trust and attribute requirements can vary widely depending which transaction is carried out for which purpose and what data are to be released and services are to be provided in the interaction. The negotiation process leverages many of the core components of the PRIME Architecture and consitutes one of the key PRIME functionalities. The negotiation is built on the human user's input as well as policies defined by the parties and ontologies.

The interaction of two parties can be characterized as follows on a very high level: A *requestor* initiates a negotiation by requesting a *resource* from a *requestee* where the resource can be a service or data. The Negotiation component of the requestee specifies, based on the request, what data the requestor must provide in order to access the resource. Then, the requestor needs to decide whether the requestee can be trusted to get the requested data. To this end, the requestor's system will construct a request for data to the requestee. This 'counter request' is created based on what the requestee has requested. The rationale behind this is that, depending on the attributes the other party wants to receive, different assurances on the trustworthiness can be required by the party. In PRIME's concrete instantiation of the negotiation

procedure the Negotiation component makes use of the Access Control Decision component to compute data requests based on the other's request.

After a successful termination of a negotiation, further interaction steps can be performed, either on the application layer or the PRIME layer. Examples for those are the user consuming a service or the user accessing their data stored at the service provider to provide updates, request blocking, or to check what data the service provider holds about them.

# 5 Component Architecture

This section specifies the component architecture of PRIME considering the major components required in a system implementing the PRIME Architecture. Note that the presentation focuses on the high-level components and does not elaborate on the internals of the components. We also abstract from implementation-specific issues that are not of general interest in discussions on the architecture level.

## 5.1 Symmetry

The user-side and service-provider-side architectures are symmetric on an abstract level, that is both essentially have the same components interoperating in the same way. The reason for this choice is that essentially 'user' and 'service provider' play to a large extent the same roles. This comes particularly clear when targeting peer-to-peer scenarios where both involved parties are users, but can be data providers and data recipients at different times in an interaction.

Regarding the symmetric of the architecture, certain functionality groups and thus components are focussed for end users, others for organizations. For example, release decision support and a simple-to-use user interface are examples for functionality for end user systems. A privacy obligation management system is rather targeted to organizations. Though, it might also be useful for end user systems in peer-to-peer scenarios where users obtain data of other users and need to handle them following the agreed data handling policy. Thus, we can see that a core of the architectural components is needed by all parties, and certain components are optionally deployed for particular classes of parties.

There is another noteworthy difference between the architecture for end users and organizations. When thinking of implementations of the architecture, there will be few or one implementation for end users as individual end users will not receive customization. For organizations, on the contrary, there are extremely heterogeneous system architectures available such that the PRIME Architecture needs to be customized towards each of those for a deployment. We want to note that for a service provider it can make sense to only deploy parts of the PRIME Architecture to implement certain functionality.

## 5.2 Overview of the Components

Fig 2 gives an overview the components of a party with the goal of communicating the main interactions between the high-level PRIME components. Note that we do not show a component responsible for secure (anonymous) communication, this functionality is largely transparent as it is subsumed in a service executed on the local machine.

A party manages a set of *Data Repositories* where each repository stores data and metadata that is held by the party where these data can pertain to this party or any other party. That is, the data subject of the data can be any party. We use this uniform approach because in our view the same protection requirements and applicable protection mechanisms apply to any data being stored. This gives a simple yet powerful architecture that is reduced to the necessary elements.

A user-side system typically will have a single data repository storing her own data (e.g., her certificate metadata or interaction transcripts) in an encrypted way. An enterprise will typically have multiple data repositories because of its data management needs.

The Data Repositories can be accessed by other components within the PRIME system and, depending on the implemented model, also by outside requestors, such as the user or a business application. Depending on this, there are different models of data storage and containment in PRIME. See Architecture V2 [MCKS06] for details. We note that full containment of data is not possible on today's systems. This may change with upcoming integrity-protected system technology. Any access is, following standard architectures for access control, done via the *Access Control Enforcement* component. The enforcement

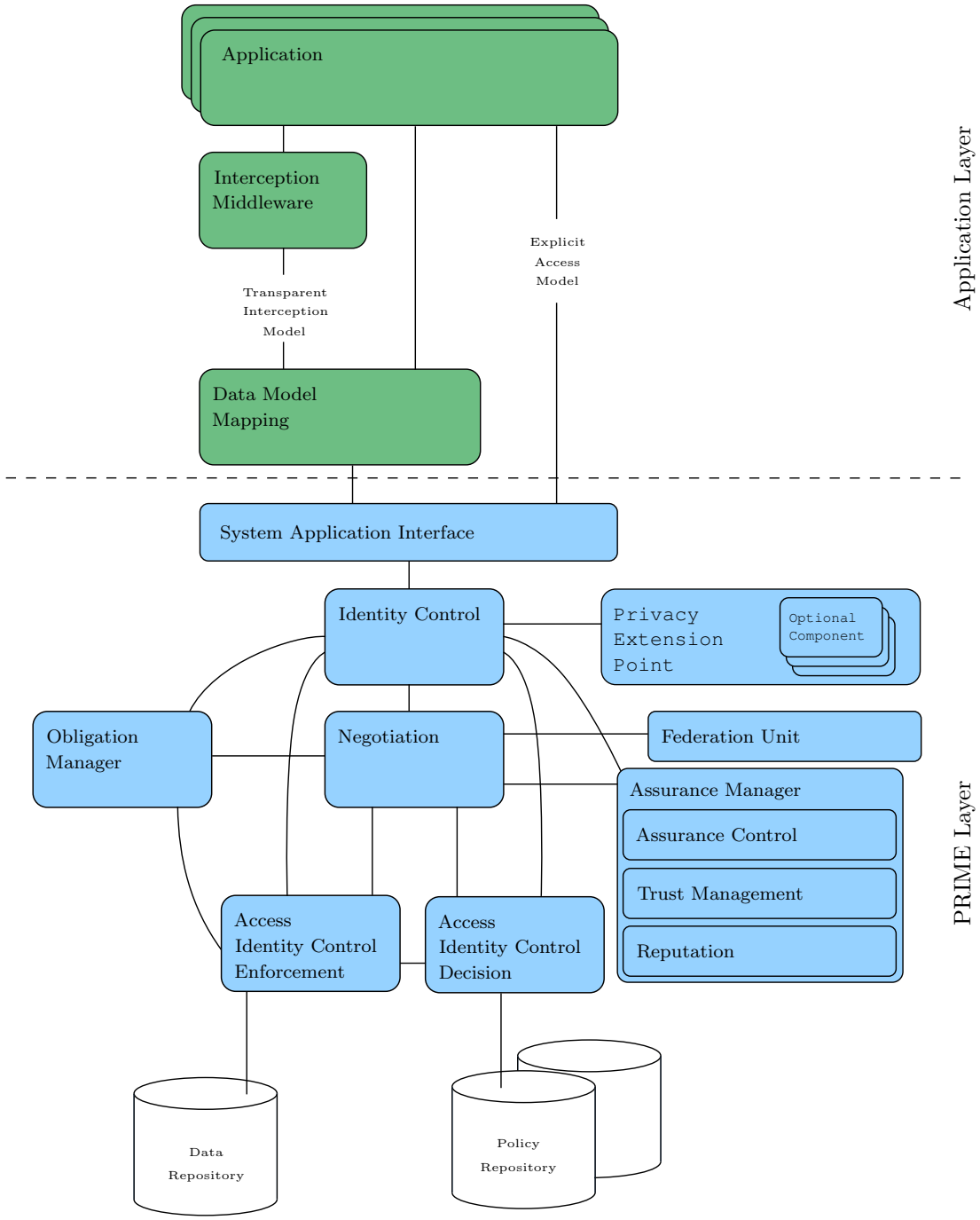


Figure 2: High-level Component Architecture

component draws on the *Access Control Decision* component to compute an access decision. Both access control components are described in more detail below.

Policies are stored in one of possibly multiple *Policy Repositories* that are then accessed by the ACD and Negotiation components to pull policies, be it access control or data handling policies.

One of the key components in the PRIME Architecture is the *Access Control Decision (ACD)* component that realizes privacy-enhancing access control decision functionality which is designed to be leveraged for implementing a negotiation protocol between two parties. The Access Control Decision component is the central component making access and release decisions on single units of data, e.g., an attribute. To do this, it evaluates access requests against the local access control policy of the party. The result of an access request is an is a grant, a deny, or a conditional with a data request. The component cannot

operate on sets of resources, such as a set of attributes to be released to another party. This abstraction is provided by the Negotiation component. ACD is stateless, requestor information that is fed in with each request is aggregated outside the component and always provided freshly as input.

The *Access Control Enforcement (ACE)* component mediates all accesses to data. It calls on ACD for an access and reads data from the data repository if access is granted. The interface is in terms of our data model described later. The component is analogous to a traditional enforcement function in access contorl architectures, with a key difference being the very advanced ACD being used for decisions. In PRIME we have combined the ACE with technology for reducing the trust assumptions on the local system, e.g., see previous versions of the PRIME Architecture [Som04, Som05, MCKS06] for details.

The *Negotiation* component implements the negotiation functionality whose primary goal is to drive the data exchange between parties, particularly in order to establish trust and exchange otherwise required data. The component receives, in each round of a negotiation, data with a data handling policy, a data request, and a proposed data handling policy for the request as input. The output of the round is data and a data handling policy and a data request with a proposed data handling policy. Thus, the input is a message $(Rq_{j-1}, Rs_{j-1})$ in round $j$. $Rs$ comprises data provided by the other party, together with the data handling policy the data is subject to. $Rq$ is a request for data targeted to the other party with a proposal for a data handling policy. The output in round $j$ is $(Rq_j, Rs_j)$ where $Rq_j$ is a new request to be issued to the other party and $Rs_j$ is the response to the request part $Rq_{j-1}$ of the previous message.[6]

The Negotiation component operates on compound requests and responses where compound means that it can comprise multiple attributes or statements over attributes. It queries the Access Control Decision component for obtaining policy decisions for atomic data units involved in the negotiation. The component aggregates the results received from the Access Control Decision component. The access of data is performed via the Access Control Enforcement component. Overall, the negotiation component in PRIME can be seen as an abstraction above ACD that provides higher-level functionality that ACD is not able to offer, but that is required in a powerful negotiation protocol.

Regarding the establishment of trust, specialized sources of relevant data can be deployed in a system. The PRIME Architecture defines a *Trust Management* component that can provide data statements to other components obtained from an assessment of the local platform, e.g., by using trusted computing technology using a Trusted Platform Module (TPM) in conjunction with an appropriate software environment. Those data statements can flow into the negotiation process, and can also be used independently of this for local platform assessment independent of a negotiation protocol with the option of issuing appropriate feedback to the user via the PRIME Console. Thus, one key functionality of the Trust Management component is to create and verify statements on the trustworthiness of platforms. See the negotiation protocol on how such information can help to establish trust between parties in an automated trust establishment process. See Crane et al. [CMP07] for a much more detailed view on the results we have achieved in terms of architecture for integrating those trust aspects into a privacy architecture.

The *Assurance Control* component is responsible for handling assurances, either for local use or in interactions with other parties. One of the features is to aggregate information obtained from the Trust Management components of one or multiple platforms comprising a services-side system. The Assurance Control component has the tasks of 1) generating assurances, and 2) evaluating assurances in order to allow the party to make trust decisions based on the assurance assessment. We prominently show the role that assurances play within the trust establishment process between two parties in the negotiation protocol in Sec. 10. For a much more detailed treatment of assurance aspects we refer the interested reader to [Pea06] paper. We can not give all details in this architecture document as it would shift the focus too far away from the architectural aspects.

The *Federation Unit (FU)* is responsible for data certification and exchange. It can implement different protocols, PRIME focuses on private certificate systems (anonymous credential systems) due to their numerous advantages in terms of privacy. The FU is the single component implementing the data exchange as dictated by the Negotiation component and the user's interventions. Legacy protocols can be integrated into this component in order to make the PRIME Architecture to be compatible with relevant protocols. The FU is comparable with the concept of a Security Token Service in the upcoming identity management architectures such as Higgins.[7]

[6]Note that in PRIME Architecture V2 the negotiation functionality was subsumed in the IdCtrl component.

[7]The Federation Unit comprises functionality of the Release and Crypto components of PRIME Architecture V2. The reasons to pull those together are to group related functionalities more closely resulting in simpler interfaces and to align better with existing architectures such as the Web Services architecture that is the basis for initiatives such as CardSpace and Higgins. The latter is particularly important when considering integration of PRIME technology with legacy identity

The *PRIME Console (PC)* is the central user interface component of PRIME. We stress the importance of the PRIME Console on the user side. It allows the system to make a user aware of what is happening in terms of identity-related interactions. It also acts an an administration interface for data and policy administration tasks. Architecturally, it is important that the Console be connected to the other parts of PRIME such that adversarial processes cannot control the PRIME system without the user noticing it. That results in the design that the PC must register with the rest, e.g., via a shared secret or PK mechanisms in order to prove its authenticity. Still, this cannot guarantee security on standard systems due to a lack of process isolation and other possibly attacks. This is an orthogonal problem.

Comparably to the user side, a Console on the server side allows for administration of the server. It is less challenging in terms of usability as it is not targeted towards the end user, but rather specialized system administrators.

The *Identity Control (IDCtrl)* component is an orchestrator for the overall control and data flow through the architecture. It executes flows within the architecture and provides facilities for the registration of components, thus providing for an extensible and flexible architecture. The IDCtrl components of two parties communicate with each other for exchanging data and agreeing on policies. Note that the Negotiation component takes over the specific parts involved in a negotiation in order to reduce complexity of the IdCtrl component. Thus, both IdCtrl and Negotiation components have can interact with most other components in the architecture due to their roles of general orchestrator and negotiation orchestrator.

The PRIME Architecture comprises some more components that have quite well-understood functinoality and are thus briefly mentioned only. Those components are 'support components' in the sense of being necessary, but not of particular interest for the privacy-related discussions. The *Audit* component logs data processing events and actions within a party that are relevant to their data processing. The *Event Management* (EM) component provides a framework for event management that is used by other components. For example, the OMS is a typical consumer of events as privacy obligations are triggered by events. The *Context* component of previous architecture versions has been subsumed in the Negotiation component as it naturally belongs there due to its task of holding the state of negotiation protocol instances. The *Crypto* component in the latest design comprises the implementation of very basic cryptographic algorithms that can be used by any other component that needs to perform cyptographic computations. Special-purpose cryptographic protocols for data exchange, such as for the execution of anonymous credential protocols, have been moved to the Federation Unit.

# 6 Data

The PRIME Architecture is centered around the concepts of data and policy and this section will be dedicated to the discussion of data. More concretely, a party holds the following:

- Data of the party itself, such as personal data pertaining to a user (attributes of the user), certificates or anonymous credentials of the user, key material of the user (private keys). Note that we can roughly classify these into attribute data and other data such as metadata and cryptographic material.
- Data of other parties, such as attribute data and profiling data that a service provider holds about its customers, or public cryptographic keys of other parties held by the party.

Note that in any case, all data items are associated to a party they pertain to, either one self, customers, or peers. This party is the *data subject*.

Within the PRIME Architecture, data about parties are *statements about parties*, made by other parties, possibly certified (endorsed) by again other parties. A typical example for such data is an attribute statement made to a service provider involving multiple attributes about a user and being certified by a European government. Data are the key objects of the overall identity management process and the key objects to be protected.

Note that statements pertain to entities, but they need not be an inherent *property* of the entity. They could rather just be any attribute assigned to the entity by any party, even the entity itself. For example, the grade assigned by an e-learning service provider to a user who has successfully completed an e-learning course, would be such an *assigned* attribute.

federation mechanisms. This is probably the best way of getting the user-centric ideas of PRIME towards the citizens.

Data need to be *represented* when being processed or stored within a party, and also when being exchanged between parties. We envision the same representation for local processing and storage as well as for the 'wire format'.

## 6.1 Basic Concepts

When considering the simplest case, an atomic item of data comprises an *ontology type* (*type*) and a *value*.[8] This data item is associated with a party. Note that most identity management initiatives as of today use this basic idea of type-value pairs to model data. When taking this obvious idea further towards better data minimization, we may define predicates over ontology types and values and use these predicates as basic data items. For example, one may state that their monthly income is greater than or equal a stated constant value, such as $greater\_than(mothly\_income\_EUR, 4000)$. A predicate thus is the atomic unit of modeling. Predicates over data are part of logical formulae that make more general statements about identity associated with a party. PRIME provides a set of predicates for making data minimizing statements, such as $=, \neq, >, \geq, <, \leq$. A predicate can operate on any ontology type if the underlying data type fits the predicate. We allow the logical *conjunction and disjunction* connectives with their standards meaning to construct formulae from predicates.

The above idea of using predicates over attributes in addition to simple $(type, value)$ pairs is a first step towards *data minimization*, that is, releasing only data that are required for a certain operation.[9] The use of the logical disjunction operator allows for further data minimization in terms of making a disjunctive statement without revealing which of the parts of the disjunction is actually satisfied.

**Example** This can, for example, be useful when a user proves to be of major age and thereby uses a disjunction of predicates each of which states the major age based on the birthdate from an id card issued by a European Union member state. The user needs to have only one id card from one EU government in order to be able to prove such a disjunction and would not reveal her nationality with such a proof.

Data items are associated with parties by means of an *identifier* for the party. An appropriate identifier is a *pseudonym*, that is, a name that is unrelated to the civil identity attributes of the party. A pseudonym is rather an identifier which addresses a party within a set of parties. A pseudonym is valid locally between two or more parties.

## 6.2 Metadata

Metadata are associated to data and give them further semantics that is relevant in the context of identity management. Prominent examples of metadata are the certifier of the data, the validity period of the certification, other properties of the certification or the certifier, how data has been obtained (e.g., method of identity federation), when data has been accessed and by whom and for what purpose, and to whom and when data have been disclosed and under which data handling policy. Some of these items are related to the concept of *trust* and *accountability*.

The method by which the technical association of data with metadata is done is not critical from a conceptual point of view.[10] Though, the way this association is done technically governs the applicability of automated derivation systems and other automated data processing on the data and metadata such as policy evaluation and enforcement.

In PRIME, we model both data and metadata in an *integrated data model*, that is, the association is inherent in the data model itself. This gives us the advantage of being able to treat metadata like any data, for example, in order to exchange metadata with other parties. Taking this abstraction even further, data and metadata can be handled in the same way technically with some limitations. One exception to this is certification metadata that might require interpretation in a data exchange protocol. Our uniform handling of data implies that policies are equally expressed on metadata, not only data. Using an appropriate representation of both data and metadata allows for handling both in a uniform way.

[8] We assume that the data type of the attribute is uniquely determined by the ontology type, for example by being specified in the data type ontology.

[9] Note that type-value pairs are *equals* predicates relating the type and the value and thus are just a special case of the more general approach of predicates.

[10] Note that the same holds true for the association of data and policies.

We think that data shall be modelled regardless of their 'ownership', that is, regardless of who the data subject is. The same enforcement and data management mechanisms should be applied to the data. In an instantiation (implementation) of the architecture, differences between those data may come up and may have an effect on choices of particular implementation-related aspects.

**Example** For example, a typical service provider keeps metadata on the enforcement history and disclosure history of data of its customers. Those metadata can then be queried by users when assessing the reliability of the service provider or the service provider can push the metadata to the user usihg the Obligation Management System, if this has been agreed in the data handling policies.

To conclude the discussion on metadata, it shall be noted that it is probably infeasible to come up with a formal definition of data and metadata that would allow to unambiguously decide whether a particular item is data or metadata. It depends on the interpretation in a specific context of whether an item is considered metadata. This strengthens our uniform approach towards handling data and metadata.

**Audit Trail** Data that are kept for audit purposes comprise an important part of the metadata of a party. This is particularly important for service providers, partially imposed by legal regulations, but also offers an important functionality for keeping track of data disclosures for users. Overall, the audit trail contains information that allows to trace where data come from, about the processing, and where they are transferred to. That is, it helps to establish *accountability*. It is also used to allow users to query data recipients on what enforcement has been done on previously released data already.

Note that arbitrary extensions can be made to the data model in order to accommodate specific audit requirements. We note that, of course, orthogonal systems for auditing can be used as well. Such systems would typically reference objects in the data and make statements about them in any appropriate data model and data schema.

## 6.3 Implications of Data Exchange

We now provide an outlook on data exchange in order to discuss its implications on the data and involved metadata. Consider a party U having an attribute 'yearly income' defined in their data repository. U can now, right away, provide this attribute as a declaration, that is, in uncertified form, to any other party she interacts with. Though, it is often a requirement that the data be certified, e.g., by the user's employer in this example. This allows U then to provide the statement to another party in this certified form, that is, the other party can rely on its correctness based on the other party's trust in the certifier, the employer.

A certification of data can be done using different kinds of mechanisms and protocols, the result is always that the recipient of certified data can have more confidence in the correctness of the data. In any method for certification, the party U requires an *identity relationship* with the employer. An identity relationship essentially represents the attributes that a certifier knows about a party.

**Example** One particular way to do a certification is to let the user U obtain an anonymous credential (private certificate) with the attribute to be certified being contained. This process includes the establishment of an *identity relationship*. A pointer to the anonymous credential is stored in the data repository associated with the identity relationship entry containing the data that are certified by the anonymous credential. This relationship can be used at the user's discretion for providing certified attribute information to other parties without contacting the certifier for each interaction. Technically, this works by the using the anynomous credential to provide parts of the contained attributes to a data recipient, still in a certified form. For example, the attribute statement $equals(yearly\_income\_EUR, 70000)$ is maintained by U in her data repository. The attribute is associated with metadata making statements about the identity relationship with the certifier, in this case the anonymous credential and associated metadata. During the negotiation process as described in Sec. 10, a policy evaluation process is performed to match this data item of the identity relationship and its metadata with a request for data from another party. To continue the example, the user can fulfill a request for $greater\_than(yearly\_income\_EUR, 40000)$ by using her identity relationship as outlined above to create an appropriate proof using the anonymous credential.

**Example** Another class of protocols for doing certification is based on protocols that involve the certifier in each data release interaction. A user U establishes an *identity relationship* with the employer and keeps the associated metadata for this. Whenever the certified attribute needs to be provided to another party, the certifier is involved and issues a fresh token including the certified attribute with the other party as intended recipient. The involvement of the certifier is done by the information in the identity relationship.

Regardless of the protocol used for certification, the data statement and most parts of the certification metadata stored by the party are the same. The protocol identifier is, of course, different. It is also important to note that the data statement being exchanged need not be the exactly same being held locally. The data statement sent to a party must be *consistent* with the statement in the identity relationship. This means that it can reveal less information, such as sending $greater\_than(yearly\_income\_EUR, 40000)$ when having the statement $equals(yearly\_income\_EUR, 70000)$ in the identity relationship. Furthermore, *disjunctions* are useful as well in order to implement the data minimization principle put forth in European legislation to an as great extent as possible. The ability to make consistent statements as shown above depends on the underlying protocol being used. It does not work for traditional certificates based on RSA or DSA signatures unless the certifier is involved in each interaction issueing a fresh security token for the statement at hand.

**The subject of data** When data are provided from one party to another, the identifier of the party the statements apply to may change to reflect the locality of the identifier and the possibility of having different *pseudonym domains*. This holds regardless of the protocol used for exchanging data. For example, two companies might know a user under a different user id (pseudnoym) as is typically the case, yet might need to exchange data about them. For such cases, the mapping must be explicitly stored, implicit in the process of data exchange between these parties, or introduced as artificial attribute.

## 6.4 Cryptographic Material

Cryptographic material is required to carry out secure data exchange protocols or secure (including privacy-enhanced) communication. Examples for cryptographic material are a public key for authenticating another party or a private key for authenticating oneself to other parties. Cryptographic material is *supportive* to the management of identities, it typically does not constitute identity information itself. A notable exception is when viewing a public key as a pseudonym of the party, which is of course an identity. For this reason, we make the clear distinction between cryptographic material and attribute data and rank cryptographic material conceptually behind attribute data in terms of importance for the concept of identity. Differently stated, for the processing of data it is irrelevant what kind of cryptography underlies their federation as long as the required trust in the data can be ensured by some means. This shows the supportive function clearly.

What kinds of cryptographic material we need for executing the envisioned identity management securely, depends to a large extent on the trust model. For example, when assuming a weaker trust model, that is, more trust in certain parties such as service providers or identity providers, we would not need certain types of cryptographic material in order to achieve the same security properties compared to a stronger trust model with less trust in parties. An example is the use of anonymous credential systems that is only required in the strong trust model as explained in Sec. 3.5 while in the weak trust model, traditional protocols are sufficient.

Cryptographic material can be stored in *secure storage devices* such as smart cards if appropriate or, less securely, in encrypted storage locations on disk. In any case, a pointer to the storage location is maintained in the correposponding data entry in the data repository to be able to retrieve the supportive cryptographic material via its corresponding data entry.

## 6.5 Open Issues

Currently it is an open question whether and how policies should be expressed for the release of statements that contain disjunctions. It might be a proper approach to move decisions on such highly compound data items onto the negotiation level and not consider them at the access control level at all. This is left to future research.

# 7 Data Model

A data model defines the semantics of data. The previous section elaborated some of the key ideas behind data, this section goes more into the aspect of representing of what has been discussed before. In many products and initiatives dealing with data, the data model is rather implicit and does not receive the attention it should receive. PRIME has worked out a data model based on the identity management requirements that the project has elaborated. The data model needs to allow different parties to exchange data about a subject in a mutually understandable form, or a party needs to be able to make automated derivations on data, such as that the attribute 'Street' is in the category 'Address' which is in the category 'Civil_Identity'. The latter can be important in order to properly enforce policies defined over attributes of type 'Address' when holding an attribute of type 'Street'.

In short, the main purpose of the data model is to constitute an *agreed semantics* that is adopted by each player in the identity management game. It is needed for core identity management tasks like for the evaluation and enforcement of access control policies and privacy obligations and for exchange of (certified) data. Abstractly, we can think of any data model as a *logical calculus* that is sufficiently expressive for the task it is to be used for. *Objects* from the domain of discourse we are interested in in identity management can be addressed by their identifiers and used in predicates in the calculus. Objects are 'things' like people, attributes, constants, or cryptographic material. Predicates can associate such objects with each other and giving a formally-defined meaning to this association. For example, the predicate 'data_statement' can associate a person identified with a pseudonym with a data statement in the form of a predicate.

So far, we have been talking about abstract objects and identifiers. On a more concrete level, an implementation of a data model in a distributed environment typically needs to address the *namespace problem*, that is, how to maintain a clean way of being able to address objects unambiguously and without naming collisions. That is, we need a way of allowing every party in a distributed environment to 1) define identifiers for objects themselves, and 2) to refer to other party's identifiers, both without naming clashes due to an inappropriate naming scheme. All of this must be possible in both a *scaleable* and *de-centralized* way.

Take as an example of a very simple data model the one consisting of unary predicates where the predicate itself models a type such as 'LastName' and the only argument models a value such as 'Doe' with the meaning that the value of the type 'LastName' be 'Doe'. This is exactly the scheme of name-value pairs as widely used today and discussed in Sec. 6. In our view such a simplistic approach is totally unsatisfactory for privacy-enhancing identity management and also for the more basic identity management. It is, for at least the following obvious reasons, too restrictive: 1) no metadata can be attached within the semantics of the calculus which is a particular problem as far as trust metadata are concerned; 2) only attribute values can be released, no partial information of attributes such as a statement that the attribute 'monthly_income_EUR' is greater than 3000 without revealing the attribute itself.

## 7.1 Abstract Model

We now express an *abstract data model* to convey the basic ideas of such a model and to give an intuition without getting lost in the formalism. Basically, an abstract model can be expressed in a calculus, such as an appropriate fragment of first order logic. Then, most of the meaning is modeled using predicates over an appropriate universe. We present important predicates and their meaning, but we do not give the complete calculus.

*Pseudonyms* are a key concept in modern identity management. Simply stated, every party can be identified by a pseudonym, regardless of whether being anonymous, pseudonymous, or identified. Thus, the pseudonym can serve as identifier of a data record as a general rule. All further attributes and predicates are associated with this identifier. For this reason, a key convention of the data model is to model data in terms of formulae, to associate those data to parties known under pseudonyms, and to associate metadata to those data. Even a party itself can express itself as a special pseudonym and associate data with it.

Next, we present a non-exhaustive list of *predicates* of our abstract data model and give their meaning.

*pseudonym*($p$): The object $p$ is a pseudonym. Arbitrary further data can be associated to the pseudonym $p$ using other predicates. This allows for representing a profile of $p$ stored at the party.

Note that a single other party can result in multiple different pseudonyms at the party without the party even knowing that those pseudonyms refer to the same other party.

me: The me keyword represents one self as a special pseudonym to allow unified modeling of the data that refers to the party itself or any other parties. One important use is to associate data to it together with metadata about the *identity relationships* the party has to make certified statements to other parties, including the association with private certificates (anonymous credentials). Those data are matched against requests in a negotiation protocol (see Sec. 10 for details on negotiation). Clearly, we allow as well that a part itself has multiple different pseudonyms to reflect different (partial) identities to be used in interactions.

$greater\_than(a, c)$: The attribute of ontology type $a$ is greater than the constant $c$ where both need to be of the same data type having a total order. The predicate can also be used to relate two attributes without revealing them. The predicates $less\_than$, $greater\_than\_equal$, $less\_than\_equal$ are defined analogously to $greater\_than$ with their respective meaning.

$equal(a, c)$: The attribute of ontology type $a$ equals the constant $c$. The arithmetic predicates can be negated with the usual meaning.

A data statement is modelled as *formula* $f$ consisting of predicates connected by logical and and or operators. A formula makes a data statement about a party. A formula can, in the simplest case be an attribute value pair, or a predicate. Multiple formulae can be associated with a single pseudonym, thus representing the *profile* associated with the pseudonym. Note that $f$ contains both data and metadata statements, that is, both are represented using the same calculus. Metadata specifically contains certification metadata allowing for trust decisions. Such a formula being associated at a service provider with a customer pseudonym expresses the knowledge of the service provider about the customer.[11]

$data\_statement(p, f)$: The data statement represented by formula $f$ is associated to pseudonym $p$, where the pseudonym represents a party. This is used to associate any kind of data statements with pseudonyms, a basic capability of the data model. Using this approach, one can represent data one knows about other parties, one can also represent data about oneself, e.g., to express what data about oneself is certified by whom to use those data in attribute exchange interactions.

$identity\_relationship(f, r)$: The formula $f$ is part of the data of the identity relationship $r$. The relationship defines the certifier, information how to contact the certifier and how to authenticate to it, what protocol the certification is based on, and other metadata relevant in this context, to some extent related to the technical realization of the identity exchange protocol(s) used with this identity relationship. Depending on the protocol for endorsing the data, the party can have obtained security tokens, and associated them with the identity relationship, that can be used in interactions with others to convey the certified data to a data recipient. A *private certificate* is an example for such a security token. This private certificate is referenced from the identity relationship $r$ and may be stored in a hardware security token.

The concept of identity relationships is comparable to the concept of *information cards* of the CardSpace model for identity federation [Mic05]. The Higgins project [Hig] uses the same model. Our approach allows for integrating the information from the card directly into the data model, thus giving it a formal semantics which the information cards lack in our view. This is particularly important when it comes to more advanced *matching* involving ontologies when performing a negotiation protocol.

$released\_data(p, f)$: Expresses the meaning that data represented by the formula $f$ have been released to a party known under pseudonym $p$. This is used to keep record on what data have been sent to whom and implements part of the audit trail. This kind of metadata is extremely useful in allowing a user to keep an overview of data releases and can be presented to a user with the *data track* function displayed via the PRIME Console.

We chose to associate the information on released data to the pseudonym directly instead of associating an audit record with the data statement of the identity relationship. The reason is that the released data statement can be one derived from the original on in the identity relationship which is not as clean to model with the other approach. A reference to the original data statement is maintained. This is required in order to allow further management operations such as automatically updating data with all data recipients once data change locally. The predicate is in direct correspondance with the concept of *partial identity*, that is, a subset of a party's attribute data. This association reflects the partial identity that the other party known under $p$ knows of the party.

[11] Note that a more elaborate method for associating data statements with parties is to include the parties the statement is about into the statement itself. This results in more expressiveness, but compromises on simplicity of the data model.

*audit_trail*$(f, t)$: Associates an audit trail $t$ with formula $f$. The audit trail contains all audit-relevant metadata for the formula, except for the data releases that are, due to their importance and for technical reasons, handled as separate entity. The audit trail particularly includes information like when the data has been created, versioning information, e.g., changes of attributes, or from whom the data has been obtained. Another important category of entries in the audit trail is enforcement metadata showing enforcement actions of the associated data handling policy. Enforcement metadata can then be communicated to the respective data subject if queried or if notification has been agreed on with the data subject in order to show compliance with the agreed data handling policies and thereby increasing trust.

*conditional_data*$(a, b)$: The attribute $a$ has been obtained through a conditional release specified by $b$. That is, $b$ contains data that allows for obtaining the cleartext attribute value of $a$ once a specified condition holds. In the case of conditional release using private certificate systems, $b$ contains the verifiably encrypted ciphertext of the value of $a$. See Sec. 8.2.2for details on conditional release.

The above gives an overview of how the PRIME data model works without covering all aspects, but rather elaborating on key aspects of the model. Specific predicates can be defined for areas such as assurance in order to communicate semantics accordingly. Also, ontology types need to be defined accordingly. In PRIME, specific predicates and ontology types have been defined that allow one to model assurances of parties to be used in the process of mutual trust establishment. As a special example, also the meaning of a privacy seal is expressed using predicates and possibly new ontology types.

**Discussion and examples** A user can maintain data for other users and these data can be as well used to associate identity relationships and security tokens, exactly like for the user's own data. This is required in *delegation scenarios* for private certificate systems when another user obtains a delegation certificate to act for a user. Those 'records' are clearly assigned to another pseudonym who is the subject they pertain to in terms of data subject rather then the acting party itself.

Whenever data are released, regardless of whether in certified or uncertified form, this is reflected by a *released_data* predicate on the pseudonym of the party the data are released to and the data being released. This new record is also 'linked' via another predicate with the original formula specifying the identity relationship used. For data statements that contain attributes from multiple private certificates, all those original records are associated with the record for the released data.

When a user releases data in multiple protocol instances under the same pseudonym to a service provider, the associated records will be maintained under the same pseudonym by the service provider. This maps exactly to the idea of a *user profile* held by the service provider. The PRIME policy mechanisms allow for defining policies on those data that may allow the user to access them, or parts of them in an interaction for making use of her *access rights* to data. This lets the service provider implement a core part of the *access to data* requirement of the Data Protection Directive 95/46/EC [Eur95] in an integrated and thus cost-effective way.

## 7.2 Relation to Policies

Both data and associated metadata must be protected by policies that govern release of data within a negotiation and other accesses. Those policies drive the negotiation process and are modeled as *access control policies* in PRIME, thus they model requirements of the recipients of the data. *Data handling policies* are associated to the data, either directly or via the ontology types, as well and specify the way data need to be handled by their recipients. All policies reside in the Policy Repository. Both classes of policies are required to fully enforce the data subject's and the legally imposed requirements on the data in terms of life cycle data management.

The association of policies to data raises some challenges to be solved. One of the more important aspects is to specify the unit of data on which policies can operate. From a point of view of data minimization, it makes sense to define this unit as a predicate, but allow as well to associate policies on complete formulae. A formula can be a disjunction of predicates and in this case it would not make any more sense to apply a policy only to a part of this formula as it would break the meaning when the policy gets enforced on parts of this formula.

## 7.3 A Concrete Embodiment – RDF

We have discussed the data model in an abstract fashion so far, without referring to a concrete calculus or even tools that can be used in a concrete implementation. Considering our discussion, the *Resource Description Framework* (RDF) [W3C04] offers a suitable calculus for our data model to be based on. RDF is powerful in expressivity as it is based on description logic. A formal semantics is available for RDF which rules out ambiguities in its processing. RDF allows for smoothly integrating data and its associated metadata into single statements and having clear formally-defined associations of those. RDF is expressed in terms of triples of the form $(subject, predicate, object)$. By referring to the same *subject* or *object* identifiers in multiple triples, a statement comprised of multiple of those triples forms a labelled graph: Subjects and objects are the vertices of the graph and subjects can be connected by edges with objects. Each edge has the *predicate* as label. A subject connected to an object via a predicate as edge corresponds exactly to an RDF triple.

Subjects, predicates, and objects in RDF are *Uniform Resources Identifiers* (URIs). This addresses the namespace requirements we have discussed for realizing an open system of having a simple, distributed, and scaleable namespace system. Overall, RDF as a calculus gives us immediately a powerful tool for realizing a data model with commonly-agreed and well-defined semantics for scaleable identity management in open environments.

Using the RDF triples for data modeling allows to model predicates and their arguments of identity statements easily. Predicates and their arguments can be modeled as multiple RDF predicates and ontology types and constants are modeled as RDF nodes. The association of such predicates with parties can be achieved by modeling parties by subject nodes in an RDF graph. We note that RDF clearly imposes certain limits in terms of expressiveness, though it provides the best widely-accepted framework with tool support for our purposes.

Any other method of modeling, e.g., an appriopriate calculus, can be used to express the data and metadata as long as our requirements for the functionality can be accounted for. For this reason we keep our discussions of the data model on a more abstract level in order to not restrict our architecture to specific technologies or calculi.

**Tools, canonicalization, and storage** One reason besides an appropriate expressivity for the choice of RDF is its tool support. *Jena* is a powerful tool for handling RDF in a computer system. It supports storage in a SQL database and powerful access and manipulation functionality. A Jena implementation is available in Java which was a good fit to PRIME's implementation efforts that are carried out in Java as well.

RDF can be canonically serialized under the condition that certain rules are adhered to. Particularly, this means that the number of *blank nodes* must be small, zero being the best choice, as the number of different canonical serializations is exponential in the number of blank nodes in the RDF graph. It is feasible to devise an RDF model that omits blank nodes yet being sufficiently expressive for most of what we need to express in PRIME.

RDF, as it is a data model, does not define or restrict the way it is stored. A prominent method is to use a SQL database for storing the RDF triples that constitute an RDF statement in a single relation. This shows the correspondance between RDF and SQL which is important for an integration in enterprise environments which usually use SQL-based storage.

## 7.4 Data Model – Mappings

We have used the idea of a single PRIME data model so far in our architecture discussions, a commonly agreed data model that is used by parties who deploy PRIME technology. Though, the typical service provider already has a data model in place, either implicit, or explicitly specified, that they use within (parts of) their organization. This requires us to define *mappings* between different data models in order to leverage the power of the PRIME data model in conjunction with PRIME technology for those service providers that operate on their own model.

Recall that a *data model* specifies the semantics of data and is agreed between the involved parties. A data model is independent of the storage representation for the data, the details of storage of data are left to a *data schema*. A prominent class for data schemas are SQL data schemas as used by most businesses for data storage. A single data model can be mapped into multiple different data schemas, and a data schema can be mapped into multiple data models. Obviously, the mappings can be defined only

in case the involved data schemas and models are sufficiently powerful and satisfy certain constraints. In order to integrate PRIME technology into the existing infrastructure of an enterprise, we need to define multiple mappings between data models and schemas of PRIME and the enterprise. We sketch the basics below and refer the interested readers for more details to [MCKS06].

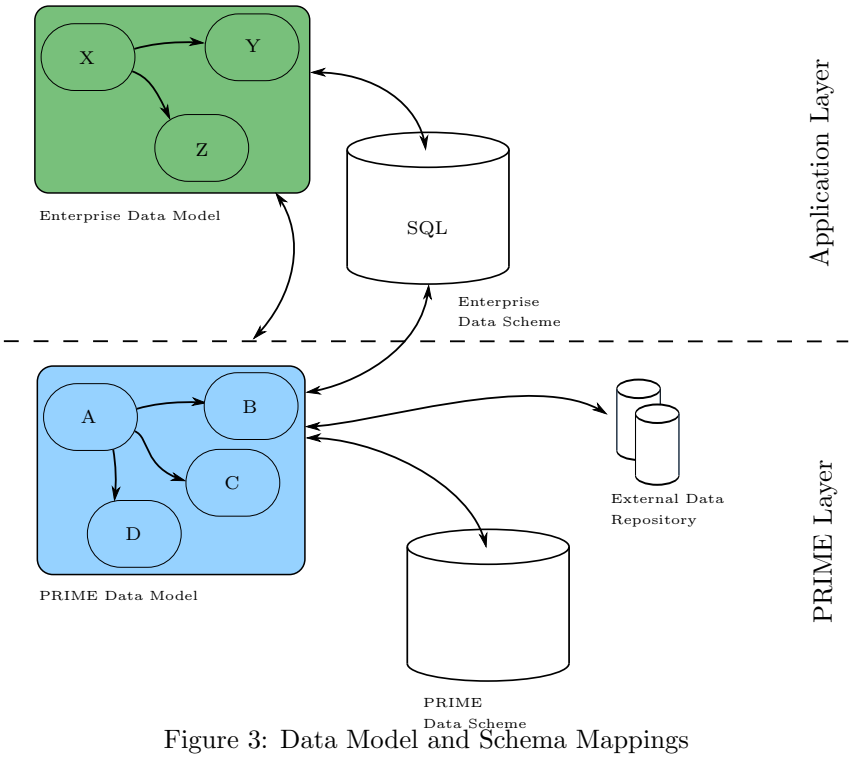


Figure 3: Data Model and Schema Mappings

Each services-side deployment may require mappings to be defined, though it depends on the specific scenario on what is needed. This is also related on the use of the transparent or explicit architectural model (see [MCKS06]). A mapping between enterprise data model and enterprise data schema is always required unless the data schema is used implicitly as data model. See Fig. 3 for an illustration of the various mappings.

A mapping between an enterprise data model to the PRIME data model allows mapping between PRIME semantics and enterprise semantics. This includes mapping between PRIME and enterprise ontology types. This is required for using the PRIME way of specifying data handling policies over the PRIME ontology on enterprise data. A mapping between the PRIME data model and the PRIME data schema is required for every party for persistent storage of data if the PRIME data schema of persisting RDF triples in SQL is used. A mapping between the PRIME data model and the enterprise data schema makes the storage and retrieval of PRIME data possible directly in enterprise databases. Note that for enterprises, the PRIME data schema is not scaleable enough due to all data being stored in one relation and the lack of performing complex queries fast. The PRIME data schema is useful mainly for users and small service providers.

### 7.4.1 Transparent and Explicit Model

The abovementioned mappings are directly related to the possibilities of having an architecture following the transparent of the explicit invocation model as discussed in more detail in [MCKS06].

In the case of the *explicit model*, applications access the PRIME middleware explicitly by using the PRIME-exposed interface. That is, the enterprise code must execute the mapping from the enterprise to the PRIME level explicitly.

When considering the *interception model*, an interception layer executes the mapping from the enterprise data model to the PRIME data model. This approach is transparent for application layer, thus the name of the model.

# 8 Data Exchange

*Data exchange* (identity federation) is the process of parties exchanging information, possibly certified by a certifier. Note that the interesting case is when the exchanged data are certified, the case of a party making declarations is rather simple. The certification of data basically 'injects' trust in the data, based on the trust the data recipient has in the certifier. The establishment of an *identity relationship* with the certifier is subsumed in the overall concept of data exchange as well. The exchange of certified identity data is a requirement in many business processes, and is particularly important if the concept of data minimization is to be followed. In many scenarios, little certified attribute data can be sufficient for a service provider to make an authorization decision, for which much more information would have to be divulged by the user in case uncertified data were be used as the service provider would typically be required to run sanity checks over uncertified data to establish sufficient trust. We want to stress that the PRIME approach is user centric, that means that the user plays a key role in the data exchange process. See [BSCGS06] for a taxonomy of user centric identity management.

Data exchange can be accomplished with different *protocols*, each resulting in different properties for the privacy protection of individuals and also different trust models in which the data exchange process is secure.

When talking about identity federation or exchange of certified data, it is advantageous to address the involved parties by the roles they play in the process. A party can take on different roles in a data exchange scenario at different times, even within a single *session*. The following roles have been identified to be required in federated identity management scenarios:

**Certifying party (certifier, endorsing party, identity provider).** A party playing this role makes statements about other parties. This often means that it vouches for the correctness of the claimed statements, but not necessarily. Liability and related issues are to be considered orthogonally to this. In business scenarios, certifying parties may assume liability for the statements they make in order to create incentives for the service they offer being used by other parties. Governments are excellent examples of certifying parties when considering various electronic ID card initiatives.

**Data provider (requestor).** A party who is a data provider makes a statement to a data recipient. This statement can be endorsed by a certifying party, if not, the data provider itself is the certifying party implicitly.

**Data recipient (relying party).** The data recipient obtains an identity statement from a data provider. Such a statement is verified to be correctly endorsed by a certifying party by the data provider in case it is endorsed by a different party than the data provider.

**Subject.** This role is a passive role in that it refers to the party that is addressed in an identity statement, that is, to whom the claimed statements apply. Often the data provider is at the same time the subject.

**Conditional release trustee.** A party having this role is supposed to provide the plaintext of conditionally released data once a pre-agreed condition holds. This role enables the conditional release of identity data by releasing the data in encrypted form or by the trustee guaranteeing to data in case of the condition being true. Anonymity revocation can be realized using this idea.

Coming back to the approach of assigning roles to parties rather than statically designating the party to fulfill a certain task in a data exchange interaction becomes clearer by considering the following examples: In the standard case of a user and a service provider establishing mutual trust during a negotiation protocol, the parties alternate in being data provider and recipient. An example for this is a peer-to-peer scenario where users make statements about other parties in the roles of certifier and data provider, and receive data in the role of data recipient. Particularly in peer-to-peer scenarios, also the role of certifier can be assumed by the parties, that is, one party makes statements about another party.

## 8.1 Protocols

**Register.** This is a protocol between a data provider and a certifying party. The data provider provides attribute information to the certifying party in order to have the certifier to make statements over these data at a later point in an instance of the Prove protocol. The data subject and the data provider need not

necessarily be the same party, but they are in most of the mainstream scenarios. Note that providing the data can be done with the Prove protocol below or with any other means, maybe by personal registration using a credential such as a physical passport or id card. The result of executing an instance of the Register protocol is an established identity relationship, possibly with associated security tokens.

**Prove.** This is a protocol between a data provider and a data recipient. In an instance of this protocol, the data provider sends an identity statement to a data recipient. The statement may be endorsed by a certifier, but need not. In the case it is not, the data provider trivially takes on the role of certifier for the data being provided. In case of the data being certified, according evidence is provided along the data to allow the other party to verify the claim on the endorsement. In a Prove protocol, data can not only be revealed in clear text, but also in obfuscated form via *conditional release* of data. The latter allows a data recipient to obtain the clear text of the conditionally released data once an agreed condition is satisfied.

**FinalizeConditionalRelease.** This protocol allows a data recipient who has obtained conditionally released data to obtain the clear text of those data once a previsouly agreed condidion holds. Conditional release is useful for achieving accountability in anonymous interactions by allowing to revoke the anonymity.

The presented protocols are abstract in a sense that they are not restricted to the choice of a particular protocol. That is, they apply to both traditional identity federation protocols as well as anonymous credential systems. The further are based on simple traditional cryptography and are secure in the weak trust model presented in Sec. 3.4 while the latter are based on more advanced cryptographic concepts and are secure in the strong trust model as sketched in Sec. 3.5. In other words, anonymous credential systems offer better privacy properties.

Next, details of the meaning of the Register and Prove protocols are given. Note that the treatment is not rigorous, but focused on the intuitive meaning as is appropriate for an architectural document.

### 8.1.1 Register

The protocol Register allows a party P to establish an *identity relationship* with a party C. C is the certifier, P the party who establishes an identity relationship with C which can later be used to make a statement certified by C to a data recipient. The parties have the following inputs and outputs in the protocol: C receives the data $d$ to be certified as input to the protocol. This can be predicates as described in our data model. Party P outputs the data and metadata $d'$ and, depending on the protocol, a set of security tokens. $d'$ contains the data $d$ and associated metadata expressed in the newly established *identity relationship*. In case of using private certificate systems, the security tokens are a private key, a public key, and a private certificate that certifies the data.

Note that for all parties a protocol transcript is output and stored as audit metadata. Both parties may be required to input key material depending on the protocol being used. Also note that the data to be certified can partially be provided by P in a Prove protocol executed before the Register protocol. In the Register protocol it is assumed that C inputs all data into the process.

The party C may want to validate the data it certifies depending on its *attribute validation policy*. The attribute validation policy may vary greatly from certifier to certifier as different use cases require different strength of validation of the attributes. A strong policy is to have attributes validated by requiring the natural person related to party P to register by personally authenticating themselves with a passport.

The parties C and P store $d$ and $d'$, respectively. C stores the data and a full transcript for accountability purposes, law enforcement, and in certain settings, for implementing conditional release. P stores the data $d'$ as an identity relationship in the data repository, and possibly obtained security tokens in an appropriate location, to use them at a later point to execute the Prove protocol on attribute information as contained in the record $d'$.

### 8.1.2 Prove

The Prove protocol is a protocol between a data provider and a data recipient. The data provider sends a data statement with attached certification metadata to the data recipient together with evidence that the integrity of the data has not been violated and the metadata apply to the data. A data statement

can contain predicates certified by different certifiers, for example, a predicate on the age can be certified by the German Government and a predicate on the credit worthiness by the data provider's bank. This powerful expressivity in terms of statements allows for a wide range of scenarios, also in peer-to-peer settings.

Depending on the concrete protocol being used, certifiers may or may not be involved in the flow. In the case of anonymous credential systems, they are not involved, but the data provider can use their private certificates that they have obtained in a Register protocol beforehand for deriving a certified statment from those.

The data provider has a set of identity relationships $\Xi = \{\xi_1, \ldots, \xi_k\}$ that can be used for the release of certified data with a data recipient. Each identity relationship contains a data statement, typically a list of $\wedge$-connected predicates over attribute types and values with associated metadata that specifies properties of the certifier and the certification protocol. Only statements $\phi$ that are *consistent* with the identity relationships in $\Xi$ can be used in a data exchange protocol. This is obvious as a certifier will only make statements about data they know about a data provider and that they may have checked in a registration protocol to be correct. We do not give a formal definition for what it means to be consistent, but illustrate the idea for the reader with an example without going into unnecessarily detailed treatment of the subject.

**Example** Let the statement $\xi$ that has been registered by a party with a certifier be $\xi = equals(last\_name, `Doe') \wedge greater\_than\_equal(yearly\_salary\_EUR, 50000)$. We do not give the predicates defining the certification metadata in this example. Let the party have other identity relationships as well. Then $\phi = greater\_than\_equal(yearly\_salary\_EUR, 40000)$ is valid as it is consistent with the statements in $\Xi$.

## 8.2 Classes of Protocols for Data Exchange

We discuss two different classes of protocols to achieve secure data exchange. Classical protocols and advanced cryptographic protocols for privacy-enhancing operation.

### 8.2.1 Classical Mechanisms

The classical mechanisms for data exchange are based on traditional signatures such as RSA and DSA to ensure authenticity and standard cryptographic mechanisms to ensure confidentiality. Two different classes of protocols exist here: 1) The data provider obtains a certificate of a set of attributes and uses this certificate in further transactions to release the contained attributes in a certified way. All those transcactions are linkable by the data recipients due to the signature in the certificate being unique with overwhelming probability and provided in each interaction. 2) The data provider obtains a fresh token from the certifier every time they want to provide certified data to a data recipient. This implies that the certifier can link all transactions, but data recipients cannot necessarily do it unless the attribute data being released allows for this.

*Conditional release* can be realized in case 2) above by having the trusted certifier guarantee to release the plaintext of the conditionally-released statement if the condition is fulfilled. This does not require any sophisticated mechanisms to be realized, but is rather based on trust.

The idea of the abovementioned approaches is to use simple building blocks to guarantee the integrity of the statement (e.g., digital signatures such as RSA and DSA) while guaranteeing privacy only under rather strong trust assumptions. Though, this is what is being used in practice today. Those methods are secure in the weak trust model as explained in Sec. 3.4.

### 8.2.2 Private-certificate-based Mechanisms

Upcoming mechanisms for data exchange are privacy-enhancing signature primitives such as SRSA-CL [CL01], BL-CL [CL02], or blind signature techniques. Those allow to construct anonymous credential systems with privacy properties that are superior to any other protocol.

The SRSA-CL or BL-CL signature schemes allow for building the most powerful systems in terms of privacy protection, so called private-certificate schemes. Those schemes allow certificates to be obtained once by a data provider and be used arbitrarily often in an unlinkable way. Unlinkability holds between each use of the certificate and the transaction of obtaining the certificate. In each use of a certifiate, a subset of the contained attribute information can be released. This does not only encompass a subset of

the attributes, but even predicates over the attributes, e.g., $greater\_than(yearly\_salary\_EUR, 30000)$ in case a predicate $equals(yearly\_salary\_EUR, 53280)$ is certified by the certificate as one of the attributes.

Blind signature schemes allow only for the construction of credential systems with the 1-show unlinkability property, that means, a credential can be shown only once, otherwise linkability of all uses is established.

The general principle underlying the use of private certificate systems is to release only what is absolutely required by the other party in each interaction. Private certificate come close to what is theoretically possible in this space. Room for improvement is still given when it comes to what we call *collateral release* of information. Collateral release is release of information that is not asked for by the data recipient, but needs to released in order to make the private certificate system work. An example is the certificate type or the issuer of the certificate that may be leaked, although it has not been asked for by the other party. A more concrete example is when the data recipient requires a proof of major age by an EU-Government-issued id card and the data provider uses her German id card. Then the nationality is collaterally released, as it has not been asked for by the data recipient. Note that proofs of disjunctions can eliminate this additional release at the cost of more computation.[12]

*Conditional release* can be realized in private certificate systems by the use of so-called *verifiable encryption*. Verifiably encrypting The resulting system for conditional release does not involve the certifier and involves the trustee only in case the FinalizeCondtionalRelease protocol is run.

Overall, private certificate systems are secure in the strong trust model presented in Sec. 3. In short, those systems offer unlinkability even if the certifier and data recipients are dishonest and share their interaction transcripts in order to link interactions of users form obtaining profiles.

# 9 Policies

This section elaborates on the policy model underlying the PRIME Architecture. The concept of policies is one of the most important concepts in the overall architecture as policies drive most of the data processing, including data exchange.

The section first elaborates on *access control policies* and discusses their expressivity and how they integrate architecturally in Sec. 9.1. This also considers the relation to our negotiation process elaborated in Sec. 10. Then, *data handling policies* and their architecture implications are discussed in Sec. 9.2.

## 9.1 Access Control Policies

One central class of policies in PRIME are *access control policies*. These policies follow the model as described in Sec. 10. Though, our architecture is more generic at some points to allow for extensibility. Our development is motivated by the fact the existing approaches to privacy protection, such as [AHK+03, PHKS02, BS02, eXt05, CLM+], are still insufficient.

In a nutshell, access control policies are defined on objects and specify the circumstances under which subjects may *access* the objects. Accessing means either consuming a service or executing an operation such as read on data. This includes data being released to (read by) another party. The access requestor can be a human or any kind of machinery for executing business processes at the party's side or another party. One key item specified by the access control policy rules is the *subject expression* that specifies requirements of a subject in order to access the protected resource.

### 9.1.1 Access Control Policy Language

We next sketch the access control policy language in more detail such that we can build on this in the explanation of the integration of the access control system with other key parts of the architecture. See [ACDS07] for details on the policy model used in PRIME. The latter is ongoing work.

**Policies** Let a party have a set of policies defined where each policy is defined as follows:

$$actions \text{ ON } object\ WITH\ object\_expression\ IF\ rules$$

[12] We do not expect that this be used in practice in the near future due to increased complexity of the protocols and the underlying policy mechanisms. We think that basic private certificates need to be established in a first step, later advanced functionality like this can be added on top of the basic systems.

The rule applies to all objects that match the object of the policy, optionally the objects can be restricted by the object_expression. The policy allows the actions as specified in *actions* to be executed on the objects as restricted by object_expression if the rules in *rules* are fulfilled.

The object can be an object identifier that can be directly mapped to an object or a concept in the ontology of the party to leverage abstractions. Note that it is possible and a typical case to refer to the object $\langle any \rangle$ in order to not restrict the object set upfront. The restrictions are then specified by the *object_expression* using appropriate predicates.

A policy is comprised by at least one *rule*. The rules in the set *rules* are to be interpreted using conjunction. Given the *object* and *object_expression*, a policy can apply to a number of objects due to the possibility of the object being a concept in an abstraction hierarchy in the ontology or any object with restrictions begin expressable by the object expression.

An *object_expression* element is a boolean formula over the object profile of an object, that is, over arbitrary structured data specified for an object. The object expression specifies the objects the rule applies to. The keyword object is used to refer to an unspecified object that is further specified via the predicates. For example, let the object be *hardware_product* and the *object_expression* by $less\_than(object.price, 100) \wedge greater\_than(object.weight, 10)$ to express that the policy applies to the ontology concept *hardware_product* that can be expanded to a set of instances of products using the ontology and to restrict the set of products to those having a price tag of less than 100 and a weight of more than 10. All products when expanding *hardware_products* with profiles specified in the party's data store that fulfill the *object_expression*, are governed by the policy defined using this object and object expression. The *object* keyword refers to the object profile. We use the '.' notation in this simplified example to refer to data stored in the profile. In practice, we can refer to any data element in the object profile when evaluating the object_expression part of a policy.

Note that not only abstractions in the ontology can be specified as object but also concrete instances of objects, e.g., the object being $product_{543654}$ which is a specific instance that can be uniquely referenced. In such a case, the object expression is typically not required to restrict the set of objects, but can be used in case one wants to express a condition on the object.

**Rules** A rule has the following form:

$$subject\ WITH\ subject\_expression\ CAN\ action\ FOR\ purpose\ IF\ conditions$$

The *subject* refers to an identifier that matches a single subject or a concept in an ontology, analogous to objects. Again, the $\langle any \rangle$ keyword can be used equally to the object of a policy. The *subject_expression* again imposes restrictions. Any of the specified subjects is allowed to perform the action indicated in *action* for the purpose indicated by *purpose* if the conditions *conditions* are fulfilled. The conditions can model any conditions, for example on the context of the transaction such as the time of the day to restrict accesses to business hours. Conditions can be evaluated on any variables and allow for powerful extensions of the expressivity of the language. The expressed conditions are a boolean expression of conditions that have to be satisfied at run time by an access request to which the rule applies.

An important property for the privacy protection is that subjects can not only be specified via a unique identity, but preferably are specified via attributes through the subject expression. This allows for attribute-based authorization semantics, being more powerful than traditional schemes while at the same time allowing for better privacy for the requester as they are not necessarily identified using their civil identity. Private certificate systems (anonymous credential systems) are the complementary technology in the space of data exchange protocols and can be used to fulfill attribute requirements expressed in the policy rules.

The *subject_expression* is essentially equivalent to the *object_expression*, with the difference that it applies to subjects instead of objects. The corresponding profiles and the expressivity are the same. Note that subject expressions are evaluated against the profile resulting from the (static) subject profile and the dynamic subject profile together, the latter being the information solicited during the currently ongoing negotiation process.

The example $greater\_than\_equal(requester.yearly\_salary\_EUR, 40000)$ requires a subject to have an attribute 'yearly_salart_EUR greater than or equal to 40000. We again use the '.' notation for referring to data associated with the subject. Note that a subject expression can be a boolean formula, also containing disjunctions. An appropriate disjunction in the subject profile can fulfill this. Such a disjunction can

result from a cryptographic data-minimizing proof made by the requester based on private certificates that shows the truth of the disjunction without revealing any further information on what predicates of the disjunction are fulfilled. This shows the relation of our policies to the advanced private certificate systems we are using quite clearly.

### 9.1.2 Release Policies

Release policies are access control policies that specifically govern the release of data. For those policies, the object and object expression refer to data items of the party and associated rules define the disclosure of these data to other parties. Those other parties are referred to by means of the subject and the subject expression, again with the full expressivity as shown above. The subject and subject expression in the rules of release policies are to be interpreted as the requirements on the data recipient if the data are to be released.

Access control and release policies are from an abstract point semantically equivalent: They protect resources and define what the requester must have provided in terms of data and what they must eventually still provide in order to access the protected resource. Accessing means either to consume a service offered by the party or to obtain data from the party. As services and data are equally modeled as resources, the cases are the same from a policy evaluation perspective. What is different is the handling in higher layers of the architecture.

When defining release policies, they not only can be defined on ontology types or instance data at an atomic level, but they also allow to talk about more general statements. For example, our architecture and policy language allows to define a policy and rule that the statement $greater\_than\_equal(yearly\_salary\_EUR, 40000)$ may be released to anyone, and another policy and rule that $yearly\_salary\_EUR$ may be released to parties that have authenticated themselves and proven that they have a privacy seal from one of a set of specified issuers. This feature of specifying policies on predicates is particularly interesting when a data-minimized statement is made in typical use cases based on an attribute value as in the example. Then the attribute value deserves a more stringent access control policy than the data-minimized predicate over the attribute value. Note that this is currently not supported in the access control decision subsystem, still we use this feature on the architectural level to keep the architecture general and compatible to future extensions of the access control function. The applicability in real-world scenarios of the above and generalizations thereof still need to be researched.

The overall architectural approach of handling the negotiation, particularly the way of driving the ACD component from the Negotiation component, allows for a powerful trust negotiation while keeping the interface of the ACD component and its policy semantics simple and implementing the higher-level part of the negotiation semantics in the Negotiation component.

## 9.2 Data Handling Policies

Data handling policies are another important category of policies in the PRIME Architecture besides access control policies. We sketch the concepts underlying data handling policies next. Data handling policies comprise two types of rules: First, rules determining *access control* behaviour for the protection of data, and second *privacy obligation* rules. Each party defines data handling policies on how they want their data to be handled by recipients and policies for defining how they will handle data received from others. Whenever a party releases data to a data recipient, data handling policies are agreed that apply to the data and satisfy both parties' policies.

### 9.2.1 Access Control Aspects

The *access control* rules of data handling policies define the access control requirements that need to be fufilled by a recipient of the data. That is, the data handling policies state, among other things, purposes for which data may be processed and classes of parties who may receive the data. See [ACDS07, ADS06] for details on the policy language that is used in PRIME. Actually, the idea in PRIME is to model the access control aspects of data handling policies using a language quite related to the access control language.

### 9.2.2 Privacy Obligation Aspects

## 9.3 Privacy Obligation Policies

Privacy obligation policies define and describe the expected behaviours and constraints to be satisfied by data receiving entities (e.g. enterprises, service providers, e-commerce sites, etc. In this section we will often refer to data receiving entities as enterprises) when handling confidential and personal data. They dictate a privacy-aware identity lifecycle management including data retention and deletion aspects, management of noti-fications and requests for authorization, data processing and transformation workflows.

Enterprises need to put in place underlying IT infrastructures, processes and mechanisms to be compliant with these obligations. This can be a challenging task due to the fact that privacy obligations can differ quite substantially given their current level of refinement (abstract vs. refined) and their "multidimensional" nature involving multiple factors and aspects.

Privacy obligations can be very abstract and generic, for example: "every financial institution has an affirmative and continuing obligation to respect customer privacy and protect the security and confidentiality of customer information" – Gramm-Leach-Bliley Act [Act03].

This type of obligations dictates high level principles and guidelines that need to be inter-preted, refined and grounded to specific contexts in order to be fully understood in terms of their operational implications. More refined privacy obligations can be expressed in terms of:

- notice requirements;
- opt-in/opt-out options limitations on reuse of information and information shar-ing for marketing purposes;
- data retention and deletion limitations.

At the other extreme, privacy obligations can dictate very specific requirements. This is the case where data retention has to be enforced for a long period of time or data are temporarily stored by organisations: privacy obligations can require that personal data must be deleted after a predefined number of years, e.g. 30 years (i.e. long-term commitment) - or in a few days if user's consent is not granted (i.e. short-term commitment). Other very specific privacy obligations might require the enterprise to notify (for example via e-mail) the data subjects, in case their data has been accessed by third parties or unauthorised people (for example in case of hacking or identity frauds). Similarly, privacy obligations might mandate to execute well defined workflows and processes, involving both humans (e.g. for explicit request for authorization) and computer systems in presence of specific events.

From a conceptual perspective, a privacy obligation can be considered as an entity (object) with a few associated properties, as shown in Figure 4.

In this view, a privacy obligation is characterised by the following core properties:

- **Obligation Identifier**: it is an identifier to uniquely identify an obligation within the entire obligation management system;
- **Targeted Personal Data**: it is a list of references to personal data that are affected by this privacy obligation. A reference must include all the information necessary to reach the data, though it can be codified in a way to avoid any indirect exposure (or correlation) of personal data.
- **Triggering Events**: it is a list of logical (AND/OR) expressions based on combinations of basic events (e.g. time, access, counters) that can trigger the need to enforce the privacy obligation;
- **Actions**: it is a list of actions to be executed at the enforcement time of the privacy obligation. Actions could be very simple - such as deletion of data or sending a notification - or much more complex, for example workflow involving both system and human interaction steps.
- **Additional Metadata**: it is a placeholder for additional properties still under exploration, such as exceptions, accountability constraints, versioning and integrity check, etc.

More formally, a privacy obligation can be seen as a $\langle i, t, L(e), C(a)\rangle$ tuple, where $\langle i, t, e, a\rangle \in \langle I, 2^T, 2^E, 2^A\rangle$:

- $I$: set of unique identifiers, associated to obligations;

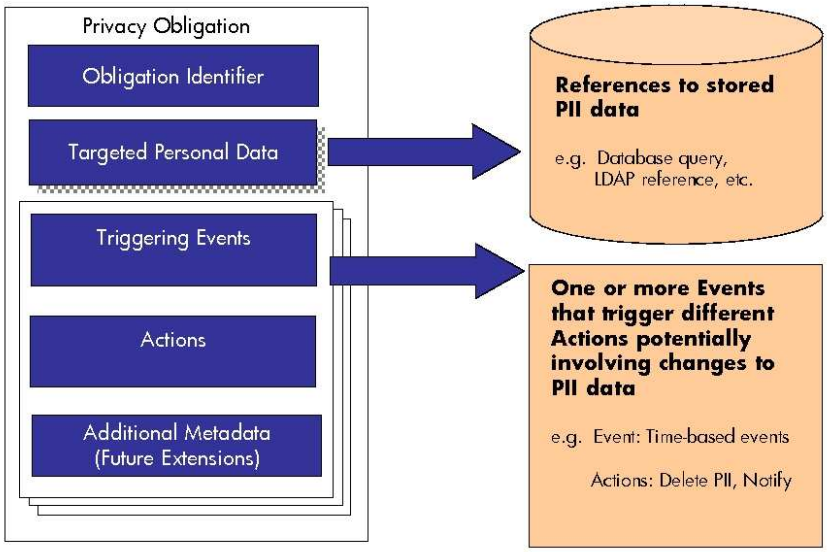


Figure 4: Model of a Privacy Obligation

- $T$: set of possible obligation targets, i.e. data entities (e.g. personal data, digital identities, attributes, etc.) subject to obligations;
- $E$: set of possible events that can trigger an obligation;
- $A$: set of all possible actions that can be executed as an effect of enforcing an obligation.

Specifically, a $\langle i, t, e, a \rangle$ tuple is defined as follow:

- $i \in I$: $i$ is an element that belongs to $I$;
- $t \in T$: $t$ is a set of targets included in $T$;
- $e \in E$: $e$ is a set of events included in $E$;
- $a \in A$: $a$ is a set of actions included in $A$.

A privacy obligation is obtained by applying the $L$ operator to the e set and the $C$ operator to the a set:

- $L(e)$: defines a logical combination of events, for example AND, OR and NOT combination of events contained in $e$;
- $C(a)$: defines an operational combination of actions, such as a sequence of actions.

It is beyond the scope of this section to provide a systematic definition or formalization of privacy obligations. In this section we will have a pragmatic view of privacy obligations, based on how we can represent them and how we can operate on them.

Note that *parametric privacy obligations* have been researched as well due to their better scalability. See ?? for details.

### 9.3.1 Making a Policy Enforceable

Once data have been released by a data provider to a data recipient under agreed data handling policies, the data recipient needs to perform two steps in order to make the data handling policies enforceable: 1) The recipient *creates access control policies* from the access control part of the data handling policies, associates them with the received data, and injects them into the Policy Repository. 2) The recipient *pushes the privacy obligation rules* to the Obligation Management System (OMS) where they get activated and subject to automated enforcement.

Step 1 above needs to be discussed in some more detail. This can in the simplest case be by just associating the access control part of the data handling policies the the data and adding the policies to the Policy Repository. In many scenarios, it would comprise mapping the appropriate agreed policies into more fine-grained data handling policies that are actually enforceable. The reason behind this is that the agreement on data handling policies operates on a more abstract level of policies that are not directly enforceable without a mapping towards the recipients specific access control requirements.

In the further discussions of data handling policies we will usually refer to both the access control and privacy obligation party collectively and make a distinction only if this is necessary, e.g., for the different handling of those parts.

pushes the privacy obligation rules of the policies to the Obligation Management System (OMS) component where the rules lead to the creation of *active* privacy obligations. The OMS then waits for appropriate events that trigger privacy obligations.

Data handling policy template; = proposal; contains multiple possible choices and parts of it must be chosen; the choice must be at most as restrictive than the proposal as expressed by the $\succeq$ operator.

## 9.4 Assurance Policies

### 9.4.1 Natural Language Examples

The following natural language expressions give examples of the type of constraints people may want to express using assurance policies:

- "the processing platforms must give a high level of protection for my PII data"
- "the back end must have a valid privacy seal issued by a provider I (or some authority I delegate) trusts"
- "the back end must give tamper-resistant protection to secrets"
- "the service provider should support obligation management"
- "my PII will only be processed within EU"

### 9.4.2 Formal View

From a formal perspective an assurance policy template can be seen as a $\langle ctid, pc, L(cc)\rangle$ tuple, where $\langle ctid, pc, cc\rangle \in \langle CTID, PC, CC\rangle$ and where $L(cc)$ defines a logical combination of cc, such that:

- $pc \in PC$: set of all preconditions
- $cc \in CC$: set of all assurance/compliance clauses
- $ctid \in CTID$: set of all unique identifiers

An assurance policy is a $\langle L(cc)\rangle$ list, where:

- $\langle cc\rangle \in \langle CC\rangle$
- $cc \in CC$ is the set of all assurance clauses

## 9.5 Relation of Policies to the Architecture

**Access control policies** A basic prerequisite for the access control decision to function are the availability of *object profiles* and *subject profiles*. An object profile stores data about an object that policy rules can apply to. Object profiles can apply to any kind of resources, that is, data and services. A subject profile contains information on a subject known under a pseudonym. See Sec. 6 for more information on the data representation. Object and subject profiles are similar except for what they apply to. In short, a profile is a set of identity data in the form of formulae, thus not restricted to pairs of the form $(attribute\_type, value)$. That is, a profile entry may be a statement like $greater\_than(yearly\_salary\_EUR, 30000)$.

Profiles can be accessed by the Access Control Decision (ACD) component and used in the policy evaluation. The policy evaluation process uses the profile information to select object a policy applies to and to decide on whether a subject fulfills the subject requirements of the policy rules. Profile information is held in the data repository. The ACD gets unrestricted access to the data repository (repositories) of the party as it is a trusted component. The actual access to the repository is handled by the Access Control Enforcement component.

For equal reasons as for object profiles, the ACD needs to read subject profiles from the data repository in order to evaluate the subject expression of policy rules. Furthermore, subject information is provided during a negotiation, referred to as *dynamic subject profile*. This has been called *context* in previous versions of the architecture, but has undergone a name change in order to avoid the ambiguity with the more general concept of context. Both dynamic profile and (static) profile of the subject are used for evaluating the subject expression part of policy rules. In many cases, the subject profile is empty, e.g., in the case of anonymous requesters.

Note that, more generally, a party can associate any information from any source to a profile, e.g., a status a customer has acquired. This information can be used in the evaluation of policy rules as well. When considering this very open approach, the importance of metadata regarding certification or origin becomes more important.

As shown in Sec. 7, a party maintains in their data repository a set of data statements that they can provide to another party in a negotiation. These can essentially all data statements associated with any pseudonym in the data repository. The most interesting case for a user-side system is the one of statements associated with a pseudonym of themselves, that is, the statements that have the party as data subject. Those statements can be accompanied with an identity relationship that indicates how those data can be provided to other parties in a certified way, or be self certified (declarations) in case not being endorsed by another party. In Bonatti and Samarati [BS02] the involved certificates and declarations are subsumed in the term *portfolio*. Our approach generalizes their notion to any kind of protocol for providing certified data. Each of those data statements in the repository will typically have acess control policies attached in order to govern their release.

**Data handling policies** We next reflect on the role of data handling policies in the architecture. This encompasses the relation to the data model and components.

The data model is designed to integrate with the concept of data handling policies, particularly privacy obligations and their implications. More specifically, the targets of privacy obligations can be any data item to retain the flexibility of associating at the appropriate granularity. Regarding the execution of privacy obligations, a focus has been put on a data model with the property that a deletion of data does not leave a trace of the underlying ontology types in the data as this could already reveal substantial information and thus compromis privacy.

The implementation of the privacy obligation enforcement is done in the Obligation Management System (OMS) component. The OMS features an extensible architecture and has a simple, clean interface. The extensibility allows for new workflows to be implemented by plugging an appropriate component into the OMS subsystem. The workflow-based architecture makes arbitrarily complex privacy obligations possible while not exposing the complexity at the interface level. Furthermore, substantial research efforts have been put into scalability of the privacy obligation management system. See Sec. 12 for details.

## 9.6 The Role of Policies in a Negotiation

**Access control policies** The access control policies as described above form the basis for the negotiation protocol of PRIME that is presented in Sec. 10. The concept is that the other party requests a

set of resources (data items or services). Based on this, the local access control policies are evaluated on the resources in the request set one by one. The Negotiation component performs the orchestration of those operations and provides the evaluation requests to the ACD. This one-by-one evaluation is required as the ACD component operates on a single resource for a single request and cannot operate on sets of resources. The policy is evaluated against the static subject profile and dynamic subject profile the party has about the other party.

The result of each policy evaluation against a single resource request is either of the following: 1) A *grant* decision; 2) a *deny* decision; or 3) a *formula* based on the subject expressions of the policy rules that apply. The results of each evaluation are composed to form a compound data request targeted at the other party. This data request can, and typically will, again comprise a set of multiple resources (data items). In case the result of all atomic policy evaluations is 'grant' no request to the other party is made and the party has authorization to release the data to the other party. One 'deny' terminates the negotiation unless the human user overrides this policy decision using the PRIME Console. Otherwise, the compound request is sent to the other party. The semantics is that this new request needs to be fulfilled by the other party before the data asked for by the other party in the original request can be released. The same process takes place at the other party in the next round. The details of this are covered in the next section on negotiation. See Sec. 10 for details.

**Data handling policies** Data handling policies, that is, both their access control and privacy obligation rules, are agreed during a negotiation protocol between two parties. The data recipient then has the duty to enforce the agreed data handling policies on the received data using the access control subsystem (ACD and ACE) and their OMS subsystem. Details this process are given in Sec. 10.

# 10 Negotiation

*Negotiation* is the process of mutual data request and exchange in order to fulfill the policies of the interacting parties. The process is policy driven and can involve humans as well by expressing their requirements in addition to the requirements from the policies. In PRIME we define the trust negotiation on the basis of two interacting parties only. Extensions to multi-party-scenarios are an open field of research.[13] The agreement on data handling policies for the exchanged data is integrated into the negotiation protocol. In PRIME, we have taken the approach of a lightweight process for agreement on data handling policies.

During a negotiation, data and metadata, the most prominent being attribute data of the parties, are mutually requested and exchanged: The data parts comprise ontology types, andpredicates over it, possibly involving constants. An example for a predicate, not showing certification metadata, is $greater\_than(yearly\_income\_EUR, 40000)$, see Sec. 6 for details on data. A special, and very common, case is to reveal an attribute value using the *equal* predicate and a constant as attribute value. Certification metadata such as the certifier or its properties, the security of the data exchange mechanism being used and other revelant metadata are accompanying the data. Using certified data can often reduce the need for further attributes, thus improving on the privacy. Though, also self-endorsed attributes can make sense in many scenarios and still dominate interactions today.

The negotiation process is executed in an automated fashion as far as possible and could even be executed without any human interaction between two PRIME Systems. Though, as the protocol needs to be compliant with the requirement of informed user consent, there is a need for involvement of the human user into the protocol, at least for giving their final consent for releasing data. The involvement of the human user into the negotiation protocol also allows for augmenting the user's system's policies by the user's requirements during the interaction. This allows for taking contextual information and user requirements into account that have not been modeled in the policy. This approach is motivated by the observation that typically it is not feasible to exhaustively model every possible circumstance that may be of relevance at some time in the policy of the party in a reasonable way. Thus, the human involvement is a powerful extension of the automated (policy-driven) approach to negotiation.

We describe the negotiation process from the point of view of a user as this is the more interesting case due to the human's involvement in the process. For a service provider, those involvement steps of

[13] Many multi-party interactions, e.g., in user-to-user scenarios can be modeled as a series of two-party interactions using our protocols. Advanced features, that inherently depend on multi-party protocols, can or course not be realized using this approach. Thus, there is an open field of research in this area.

the human need to be omitted.

## 10.1 Preliminaries

We next give necessary preliminaries that help understanding the negotiation process better and at the same time introduce some of the notation used in the explanation of the negotiation protocol.

**Data handling policy** First of all, recall that data handling policies contain privacy obligations, that is, provisions that must be fulfilled by a data recipient and which may be completely independent from access control aspects. The remaining parts of the data handling policy are related to access control. We always subsume, unless explicitly stated otherwise, both those parts when talking about data handling policies.

The data handling policy negotiation in PRIME is simple in order to not make the overall negotiation protocol unnecessarily complex. The basic idea is that the requester of data makes a *proposal* $\pi_{j-1}$ for the data handling policies for the requested data $d_{j-1}$ that leaves certain parameters variable. We call this proposal also *template*. The respondent makes a choice, including instantiation of the template $\pi_{j-1}$ with concrete values in a proposed range and the selection of certain options resulting in the *agreed data handling policy* $\pi'_j$. The key property of this constrained approach is that the new data handling policy must not be stricter than the one proposed. This property ensures that the provided policy will be acceptable by the party who requested the data and proposed the template policy.

The difficulty of having more generic and open-ended negotiations of the data handling policies is that in traditional scenarios between a user and a service provider, the service provider has the need of *regulatory compliance* with a plethora of legal requirements and cannot accept any data handling policy of users, but needs to stay in some, often tight, range. This means that the negotiation space is quite restricted upfront in terms of service provider requirements. Key aspects of the data handling policy that can be more freely negotiated are ones that do not affect regulatory compliance of service providers, e.g., to notify the in case some of their data are provided to a third party. Considering peer-to-peer scenarios, we face a completely different situation as there are currently no regulatory requirements applying to users who gather data of other users. This makes users who negotiate with each other more free in negotiating their data handling policy, that is, for such scenarios a more general negotiation of the policy is useful. Our approach is extensible to such scenarios and multi-round data handling policy negotiations.[14]

For performing a check whether the data handling policies fulfill a proposed data handling policy template, a comparison calculus is needed. We use the following notiation: $\pi'_j \preceq \pi_{j-1}$ means that the policies $\pi'_j$ are at most as strict as the template $\pi_{j-1}$. This is the case when the data handling policies $\pi'_j$ are compliant with the proposed template $\pi_{j-1}$ in the sense of not requiring more than any instantiation of the template. Only this case constitutes a successful agreement on a data handling policy between a data provider and data recipient.

**Data request and statement** Let $d_{j-1}$ be a data request and $d'_j$ be a data statement. We use the operator $\sqsupseteq_d$ as follows: $d'_j \sqsupseteq_d d_{j-1}$ if and only if the data statement $d'_j$ contains at least the data that are requested by the request $d_{j-1}$. This considers ontologies, for example, for expressing the certification properties of the data and ontologies over the attribute types. This informal definition suffices for our purposes of describing the negotiation protocol.

A *request* $d_j$ is a formula $f$ as described in the section on data representation with the additional extension of having two different meanings of the $\vee$ operator in terms of the request. 1) The party is required to fulfill one term in the disjunctive normal form of $f$. 2) The party may provide a proof that the overall formula is fulfilled without revealing which of the individual predicates are fulfilled. Option 2) is preferable in terms of data minimization as the party can hide which predicates it can fulfill while fulfilling the overall formula. Note that this requires either sophisticated cryptographic mechanisms that allow one to compute proofs of disjunctions, or 'proving' the formula by involving a trusted party as certifier that vouches for this formula. The latter operates in a much worse trust model as the certifier learns much more about the interactions the prover performs, but can still be used to implement this functionality.

[14] In our view, itn is an interesting question whether the regulatory vacuum for users who process data deserves initiatives from the side of lawmakers, particularly as users can, considering today's data processing equipment, handle masses of other user's data due to their various interactions in collaborative activities over the Internet.

The general motivation for supporting the $\vee$ operator in a data request is to allow a party to provide multiple different options for authenticating a requesting party and thus authorizing a request. Then it is sufficient to fulfill the disjunction which can be done either by providing a subset of the requested information such that the formula is fulfilled, or proving, e.g., in zero knowledge, that the formula is fulfilled as outlined in the previous paragraph. The following example illustrates this: One can log on to a Web site with a $k$-show service subscription credential, or use an unlimited subscription, or log in to a pseudonymous account, or log in to an account where the user is identified. Depending on what is chosen, different degrees of service customization might be available to the requester.

The case of $d_{j-1}$ containing $\vee$ operators can in its general form be as complex as outlined above. A typical user is not cabable of grasping the semantics of a formula containing $\vee$ operators in an arbitrary fashion as the intuition does not allow one to grasp the meaning. In this case, a normalization to disjunctive normal form or a clever, yet to be researched, way of presentation interface might make it feasible to be decided by the user. Presenting a complete recommendation to the user and just soliciting user consent on it is probably a preferable way in the general case as it takes the resulting complexity away from the user. Though we think, that it is not a viable task to define policies that allow for automated decision support in this general case.

An *data statement* $d'_j$ follows the representation for expressing data statements as used in PRIME. See Sections 6 and 7 on data and the data model for details. Recall that a data statement $d'_j$ expresses data as well as certification metadata semantics in order to allow the data recipient making authorization decisions based on it. More concretely, the message contains predicates that are connected with the logical connectives $\wedge$ and $\vee$ talking about both the data and metadata. A data statement is provided by a party in response that has been previously issued by the other party in a negotiation.

**Subject profile** Each party maintains a *dynamic subject profile* $K'$ for keeping track of the data having been received from the other party in the current negotiation. An already populated dynamic subject profile can be reused in a later negotiation within the session to avoid re-providing data multiple times and thus increase efficiency of the protocols. Recall that a party can have a *(static) subject profile* $K''$ as well containing information about a subject. The static profile is exactly the information about the other party stored in the Data Repository.[15] Let the *compound subject profile* $K$ be $K := K' \sqcup_K K''$ where the $\sqcup_K$ operator merges two profiles.

A compound profile $K_A$ of a party $A$ fulfills the set of (access control) policies $P_A$ of a party $A$ for a compound request $d_{j-1}$ for the purpose of releasing the data to the party the profile pertains to if the data statements in the profile let all authorization policies that apply to each requested data item of $d_{j-1}$ be fulfilled for all requested data items. See [ACDS07] for details on the policy evaluation semantics for a request to a single resource. We denote the function for performing an evaluation of such a compound request $\mathsf{eval}(P_{A,j}, d_{j-1}, K_{A,j})$. For the evaluation of the individual authorization policies on a single resource, a *read* operation is assumed to be the operation to be executed as this is the operation that applies for releasing data in a negotiation.

**Notational conventions** We use the following notation: In round $j$, a message $m_j$ is constructed and sent by the party where $m_j = (Rq_j, Rs_j)$. The message part $Rq_j = (d_j, \pi_j)$ is the request part of round $j$ comprising data $d_j$ and a proposal for the data handling policies $\pi_j$ for the data $d_j$. $\pi_j$ is a customizeable data handling policy template that can be customized (instantiated) by the other party following their data handling policies and possibly human requirements. The customizing process results in a data handling policy. $Rs_j = (d'_j, \pi'_j)$ is the response part of round $j$, being a response to a previous request part $Rq_l = (d_l, \pi_l)$ of message $m_l$ of the round $l < j$. The response comprises data $d'_j$ satisfying the request $d_l$ and the data handling policy $\pi'_j$ that is at most as strict than the proposed policy $\pi_l$, that is, $\pi'_j \preceq \pi_l$. The reason for using a combined request-response message is to allow for providing data ahead of a request for efficiency reasons for common data requirements in order to reduce total network latency.

**A round of negotiation** A round $j$ is defined by a party $A$ receiving a message $m_{j-1}$ from the other party $B$ in the negotiation, performing computations dependend on this message, and sending a message

[15]The profile is in PRIME addressed by the pseudonym it is associated with. For many scenarios it makes sense to link the current request to the static profile and make use of the information contained in the static profile as well for negotiation. This is particularly true, if the static subject profile contains attributes that have not been provided by the user, but otherwise obtained or derived.

$m_j$ to the other party. We say that $A$ *executes* this round. In each round one party performs computations and parties alternate in executing rounds until the protocol terminates.

**Start of a negotiation** In the PRIME negotiation protocol, it is envisioned that a negotiation always starts with one party $A$ requesting a resource or set of resources of another party. Each resource can be, as usual, a service or data. This includes the situation of a party starting with a request for data which is interesting for the case of a user requesting assurances from a service provider before even engaging in further interactions. The specifics of a round 1 that constitutes the starting round of a protocol is that the player executing it only generates a message $m_1$ without running through the computation protocol for the round and without receiving input.

From a cross-layer perspective of the application layer and the PRIME layer, a negotiation can be started either by an action on the application layer or an action on the PRIME layer. Conceptually, those cases are the same as in either case, control is passed to the PRIME layer and, after executing the negotiation protocol, control goes back to the instance on the triggering layer. An example for triggering the negotiation protocol from the application layer is the access of a resource of the other party by the user's browser. This triggers the interaction on the PRIME protocol layer. An example for triggering the negotiation protocol from the PRIME layer is when the user or her PRIME system requests assurance data from the other party.

**Termination of a negotiation** There are four possibilities of how a negotiation can terminate: 1) a pre-defined threshold $k_{\max}$ is reached; 2) one of the parties explicitly terminates the negotiation using a specific termination message $\tau$; this includes the case of the party not being able to satisfy the request of the other party; or 3) the access control policies $P_A$ and $P_B$ of both parties $A$ and $B$ are fulfilled with respect to the parties' compound subject profiles for the other parties leading to a successful negotiation. Note that case 1) and 2) may result in a situation in which parties already have released data to each other, and still the protocol terminates without success.

**A typical example** We anticipate that in typical interactions between a user and a service provider, the negotiation protocol would consist of 5 rounds, as shown next without going into details: 1) A user requesting a service; 2) the service provider sending an assurance statement and data request to the user; 3) the user requesting further assurances from the service provider; 4) the service provider satisfying the user request for further assurances; 5) the user providing the data requested by the service provider in round 2). When using the tricks of running an anonymous credential system as non-interactive protocol and biggy-backing the required challenge in a previous message, the credential protocol is executed in the negotation round 5) by the user and the resuling message sent with $m_5$, thus no additional round is required for the privacy-enhancing data exchange protocol. In the final round, the protocol terminates with success and the user gets access to the initially-requested service.

## 10.2 Protocol

This subsection describes how one round, round $j$, of a negotiation protocol between two parties $A$ and $B$ is realized in terms of computations and data flows within the party' PRIME system. We remind the reader that parties take turns in executing rounds until the protocol terminates with or without success. Note that for round 1 the below does not apply as stated, but only a request is sent to the other party. This request can technically be in a different form such as a request by the party's Web browser by clicking a link associated with a resource. We describe the round from the view of party $A$.

**Reception of input message** $m_{j-1}$ The message $m_{j-1} = (Rq_{j-1}, Rs_{j-1})$ is received by the party. An exception is round 1, the first round in a negotiation. Recall that $Rq_{j-1}$ is a tuple $(d_{j-1}, \pi_{j-1})$. In case $m_{j-1} = \tau$, the protocol is terminated without success and no further message is sent.

**Storage of received data** If the message contains data $d'_{j-1}$ and associated data handling policies $\pi'_{j-1}$, the following processing is performed: 1) The data handling policies $\pi'_{j-1}$ are checked whether they correspond to the template proposed to the other party in a previous round when requesting the data. Only if the check succeeds, the party proceeds with the processing. 2) Depending on the data handling policies (or by the application data are collected for), the data are stored in the data repository in the

static subject profile pertaining to the other party.[16] All received data are added to the *dynamic subject profile* $K'_{[A,j-2]}$ associated with the other party resulting in a new $K'_{[A,j]} := K'_{[A,j-2]} \sqcup_K d'_{j-1}$. The data are used already later in this round in an access control policy evaluation. The data handling policies are handled depending on their type: The part of the data handling policies that is related to access control is used to *derive local access control policies* that are stored and associated with the data they are supposed to apply to. Those policies go into the Policy Repository. This allows for a later enforcement of the access control policies by the Access Control Enforcement component and Access Control Decision component. The parts of the policy that are related to privacy obligations are handled differently. Every privacy obligation with its association to the data items it applies to is pushed to the Obligation Management System.[17]

The request $Rq_{j-1}$, if not empty, is pushed on a stack $\Theta$ of requests that is valid throughout this negotiation. The idea behind the stack is to use it to always operate on the latest request (topmost element) and work downwards. Let $Rq_\eta = (d_\eta, \pi_\eta)$ be the topmost element of the stack.

**Computation of a response data statement** In the next step $B$ needs to decide how the request $Rq_\eta = (d_\eta, \pi_\eta)$ will be fulfilled. The intuition behind this step is that a data statement $d'_j$ is created using both identity relationships for certified data and uncertified data entries such that $d'_j$ fulfills $d_\eta$. There are two basic approaches on how to do this: a) Making a complete decision on the satisfaction of the request in one step of computation and user interaction or b) having a 2-stage process with first deciding on which of the predicates of the request to fulfill and then deciding on the identity relationships to use to fulfill the chosen parts. Method a) is preferable as it can compute a more optimal decision in terms of privacy protection compared to method b) unless method b) offers a possibility to reiterate the first stage of the 2-stage process, in which case, the possible solution spaces of both approaches are identical. Note that $d_{j-1}$ might contain $\vee$ connectives with one of the two meanings as outlined further above.

In either of the cases a) or b), the final outcome of the decision process is a data statement $d^*_j$ that will be part of $Rs_j$. This data statement is an unambiguous specification for how the request $d_\eta$ can be fulfilled. The data statement $d^*_j$ forms the basis of what will be provided to the other party at a later stage in the round or a later round.

Optionally, $d^*_j$ can be *extended* with data that the party $A$ expects to be useful for the other party $B$ in assessing trust of $A$. The most important example for this are *assurance statements* that would be provided by a service provider in round 2 of the protocol (after a user has requested a service). An assurance statement would be pulled from the Assurance Control component. The final resulting data statement is $d'_j$. When omitting this optional extension, the party sets $d'_j := d^*_j$.

The statement $d'_j$ specifies, like $d^*_j$, unambiguously how a *Prove protocol* (see Sec. 8.1) with the other party $B$ can be executed using *identity relationships* the party $A$ holds. Note that $d'_j \sqsupseteq_d d_\eta$ must hold, that is, the response fulfills a previous request.

**Construction of $\pi'_j$** Next, the data handling policy $\pi'_j$ must be constructed in a way that it fulfills the template $\pi_\eta$ previously proposed by the other party. This is done by matching the proposed template with the party's own data handling policies from the Policy Repository storing the data handling policies $\Pi_A$ and allowing the human user to make choices as far as allowed by the template policy. This particularly includes privacy obligations. In case of a mismatch with the own data handling policies, the user is given notice of this and the choice between overriding the local policies $\Pi_A$, either for this instance of the negotiation protocol, or permanently, or terminating the negotiation. The relation $\pi'_j \preceq \pi_\eta$ must hold for the finally constructed policy $\pi'_j$.[18]

**Computation of a data request** Next, the party $A$ needs to compute a data request $Rq_j = (d_j, \pi_j)$ targeted at the other party. The request is based on both $A$'s data release (access control) policy and the data or, more generally, resources, that the other party has requested in $d_\eta$. For each predicate $R_{[\eta,i]}$ of the request $d_\eta$ of *sufficiently simple form* (see below), the party's Access Control Decision component (ACD) is invoked in order to query whether $R_{[\eta,i]}$ may be released, that is, whether a *grant* decision can

[16] For a party that is not known yet, this can lead to the creation of a new subject profile.

[17] Note that in addition to the data handling policies that have been agreed with the other party, policies can be define within the party that apply to the newly solicited data. Those policies would typically be defined over ontology types and automatically apply to all data of those ontology types.

[18] For assurance statements a service provider will typically assign a data handling policy that does not impose any use restrictions as there is no harm in this information being used arbitrarily.

be obtained, or whether it cannot be released (*deny* decision). A third possible result from the invocation of the ACD can be a data request for data that are additionally required from the other party before $R_{[\eta,i]}$ can be released. Let the individual results of the invocation of the access control on $R_{[\eta,i]}$ function be the partial request $d_{[\eta,i]}$. $d_{[\eta,i]}$ can be a data request or a *grant* or *deny* decision.

The requirement for predicates to be in a sufficiently simple form is constrained only by what the policy language can accept as objects. Thus, it depends on the instantiation of the policy language and is subject to change with the expressivity of the language. Note that for general predicates it is currently not clear on how the policy could handle those in a practically viable way to derive a counter request from such a formula. This is open to future research. A key requirement is that the policy definition process must remain manageable. This becomes clear when thinking of predicates that relate multiple attributes with each other or other special predicates for which no specific policy rule will reasonably be available. Simple cases that can be handled by the policy are, for example, predicates that request an attribute or predicates that request a comparison relation over an attribute and a constant such as $monthly\_salary\_EUR \geq 3000$. The latter case is prominent when considering PRIME's capabilities in terms of data minimization.

*Compose* the individual requests $d_{[\eta,i]} \notin \{grant, deny\}$ resulting from the above invocations of the Access Control Decision component to obtain $\hat{d}_j$.

Next, the human user is involved. If there was a single *deny* result among the $d_{[\eta,i]}$ elements, this is presented to the human user in order to give them the possibility to make an exception and still progress the negotiation. This allows the user to override her access control policy $P_A$ for a particular interaction or permanently redefine it.[19] At this point the negotiation will be terminated if the user does not override the policy in case of at least one *deny* authorization result. Termination is done by the party sending a $\tau$ message to the other party.

Another part of the user interaction is an optional extension or reduction of $\hat{d}_j$ by the human user to reflect additional requirements not captured in the policy. At this point, the user can extend $\hat{d}_j$ such that further requirements from the user's side not expressed in her policy can be taken into consideration in the negotiation. The extensions here typically involve additional assurance-related requests not yet covered by the access control policy, particularly details or evidence for some assurance statements. The Assurance Control component can provide support on additional assurance requests. A proper user experience for those additional data requests requires an appropriately crafted user interface for handling this. Other components could do an automated post processing as well on $\hat{d}_j$ to account for other requirements. The updated request is $d_j$.

**Creation of $m_j$** When all access control decisions on items of $d_\eta$ by the ACD component have yielded *grant* decisions, that is, if $\mathsf{eval}(P_{[A,j]}, d_{j-1}, K_{[A,j]}) = \mathsf{true}$, the party sets the first element of the response part $Rs_j$ of the message $m_j$ to $d'_j$. The party then also sets the second element of $Rs_j$ to $\pi'_j$ to account for the data handling policy template.

If $d_j \neq \epsilon$, the party sets the request part $Rq_j$ of $m_j$ to $Rq_j = (d_j, \pi_j)$ where the data handling policy templates $\pi_j$ for the request are pulled from the Policy Repository. The element $\epsilon$ is the empty request.

**User consent** The user is presented a consent view. This view can be integrated with the previous functionality for identity and policy selection. The user gives consent in order to trigger the data release. The user can terminate the transaction at this point. A termination results in sending a $\tau$ message to the other party.

**Sending of message $m_j$** At this point, the request $Rq_j = (d_j, \pi_j)$ gets issued to the other party within the message $m_j$.[20] The message includes the previously computed data statements.

Sending the message $m_j$ includes the *release of data*. The data is provided in the element $Rs_j = (d'_j, \pi_j)$ of the message. Releasing the data means that, if this is required by the specification, a *proof*

[19] This is analogous to the case of the data handling policies above. Note that the approach of realtime policy definition is to be taken for those cases from a user interface perspective, that is, the user can within the negotiation redefine her policy in the appropriate context.

[20] The other party will (potentially) fulfill the request later by providing data. This is completely symmetric to the negotiation step described in this section, with the difference of no user involvement in case of the other party being a service provider and not a peer. In the case of a user negotiating with a service provider, we envision that the service provider does not ask the user for further data in a typical negotiation. Though, further negotiations may be executed later when other services are consumed.

needs to be generated by the party following the specification, or the party needs to request a proof token from an identity provider, or a combination of both is applied.

Sending data involves for many cases the *Federation Unit* component to execute an appropriate protocol for data exchange. In the case of assurances being provided, the appropriate protocol is executed by the *Assurance Control* component. The Trust Management and Reputation Management components are other possible executing engines of protocols or sources of evidence tokens to be sent to the other party. Further complications come into play here as multi-round *interactive protocols*, such as zero-knowledge proofs of knowledge, may be required to be executed for conveying the required evidence for the data statement in a privacy-preserving manner as envisioned by PRIME. All those protocols are executed as standalone protocols within the negotiation protocol and can result in multiple round trips just in this single negotiation round. Tricks like piggybacking required (cryptographic) challenges in previous negotiation messages and the use of non-interactive cryptographic protocols may decrease the number of round trips, but needs to be carefully analyzed due to potential interdependencies of the protocols and resulting non-composability.[21]

**Update of $\Theta$** If all $d_{[\eta,i]}$ have been *grant*, remove $Rq_\eta$ from the stack. In this case, the party has been able to fulfill the request $Rq_\eta$ and can, in the round after the next, tackle the next request that is still on the stack.

### 10.3 Efficiency and Rationale

**Efficiency** The design goal of our negotiation protocol is to keep the number of rounds low as this has a great effect on the efficiency in terms of runtime and thus acceptability in practice. Depending on the mechanisms used, the majority of the runtime is caused by network latency. In the case of private certificate systems being used as mechanism for privacy-enhancing data exchange, the computational load is increased in relation to latency. We have given a brief discussion on efficiency aspects in this case above. Industry adoption of protocols typically requires protocols having as few rounds as possible.

**Protocol Design Rationale** The justification of our approach for the selection of data to release and building up the counter request is based on the fact that different data items require different protection by their potential recipient, that is, information about the other party of different types. This information gets requested by the party in order to get confidence that the other party will behave as it should. Of course, this trust establishment is no guarantee whatsoever that the other party will behave as advertised, though, it is practically useful evidence, in particular if it includes statemenst endorsed by external party, e.g., a privacy seal. Concludingly, we do not live in a perfect world in terms of no party ever failing either intentionally or unintentionally in living up to their data handling promises. For this purpose there is still a legal framework with law enforcement in place as a last resort which is here for handling problem cases.

## 11 Assurance Checking

This section discusses assurances and their role in establishing trust between parties, particularly between a service provider and a user. The discussions in this section complement the discussions of the interplay of assurance mechanisms with the general negotiation process.

Assurance policies express the security and data protection processes and mechanisms which should be in place to protect users' data. Users define such policies to express the minimum privacy protection which they wish to have in place by the recipient of their data. Service providers publish their policies to assert protection which should be in place, and for which they are able to provide evidence that this is indeed the case. Thus, assurance policies may be regarded as a specialised form of release and data handling policy, depending upon the context.

In this section we explain the motivation for using assurance policies, and show some formalisms used within PRIME.

[21] Further consideration of these matters goes far beyond the architectural level and will thus not be further elaborated on.

## 11.1 Introduction

Assurance checking policies are formulated by people to obtain degrees of assurance from enterprises that their data will be processed according to their expectations, such as compliance to privacy, security and IT standards. In many cases, the user is specifying the type of device and environment where their PII data is being viewed. These would usually be checked up-front before PII was released, either in the preamble or in a negotiation phase before release of PII.

Assurance checking policies are separate from obligations and are constraints and conditions usually expressed by people before they engage with enterprises. They might just specify a particular regulatory context or, more generally, can require enterprises to provide degrees of proof about their ability to:

- Support the enforcement of predefined privacy policies and obligations with respect to laws and legislation
- Run their processes, services and data repositories in a secure way
- Use secure and trusted systems, such as trusted computing platforms, to increase the level of security and trust in their operational activities.

Assurance checking policies may also be expressed by the services side, to obtain degrees of assurance from third parties that any data shared with these third parties will be processed according to the service provider's expectations, and also to express how data provided to the service provider by users will be processed, and to provide evidence that this is actually the case.

In summary, the overall motivation for defining assurance policies is to allow people to make judgements about the trustworthiness and privacy compliance of the remote receiver of their PII data. For example, a user might check for the compliance of an organisation against customised preferences prior to disclosure of PII data. Their disclosure of PII data or the continuation of a business interaction could be subject to the outcome of this checking.

### 11.1.1 Principles

The principles underlying the use of assurance policies are as follows:

- The assessment goes beyond just promises on behalf of the service provider
- It assesses the levels of proof that can be provided about privacy-providing mechanisms that are used and even how they are operating
- A broader range of information and assurance is assessed than just security information
- In some cases assurance policies can be checked independently of access control
- Broadly speaking, the conditions checked are necessary, not sufficient for the transaction to proceed
- Ultimately, it is the user who decides how to proceed

See Sec. 9.1 for examples and an informal definition of assurance policies.

### 11.1.2 Overview of Different Potential Approaches

There are a range of different approaches to defining assurance policies. We shall consider two approaches, both of which we considered within PRIME:

1. Trust and assurance constraints can be represented as first order predicate logic expressions: for example, `hasValidPrivacySeal(Issuer) & isTrusted(Issuer)`. These would be conjoined to other conditions within the 'conditions element' of data handling policies, transfer policies, etc. In particular, we extend the access control predicate vocabulary given in Section 9.1 by defining a set of assurance-based predicates of the form `predicate_name(`*arguments*`)`. In this case, the assurance control checking is invoked as part of the Access Control Decision assessment within PRIME. Here, the assurance policy can be injected into the existing access control policies as predicates. For example: `IF access Control checks) AND IF (Assurance Control checks)`

2. An alternative approach is to use a much higher-level representation in which individual human-readable clauses within the policies are first class objects, thus defining a completely separate policy representation language from the other approaches presented in this chapter. Here, the checking is invoked as an independent assurance control function invoked by an entity to conduct tests against a compliance template: for example, a user initiating compliance checking of a website against their "e-commerce" policy.

Accordingly, there are different possible levels of representation that can be used within assurance policies, respectively:

1. At a low level e.g. $\forall x \in$ `ProcessingSystem (hasWorkingTPM(x)` $\vee$ `...)  & usesAdequateEncryption(x) & isPatched(x) & hasWorkingOMS(x)`, with reference to a lower level semantics

2. At a high level e.g. `checkTrustedProcessingSystem`, with reference to a higher level semantics

The result of the assurance checking is an analogous structure to the assurance policy input (with the resulting values of the assurance clauses). The complexity can be shown to the user if desired, and the structure can simplify to a Boolean value, for instance so that the Access Control Decision point may make a decision about an authrization request. The latter shows well the tight integration with the overall architecture, particularly negotiation and access control.

The following sections consider these different approaches in turn.

## 11.2 Defining Trust Constraints: A Lower-level Representation

Our initial Prime implementations used the notion of trust constraints [Pea06]. Trust constraints are part of a broader representation of constraints within policy languages. Figure 5 illustrates how, within the context of policies and preferences, they are a subset of a broader set of constraints about data processing: assurance constraints.

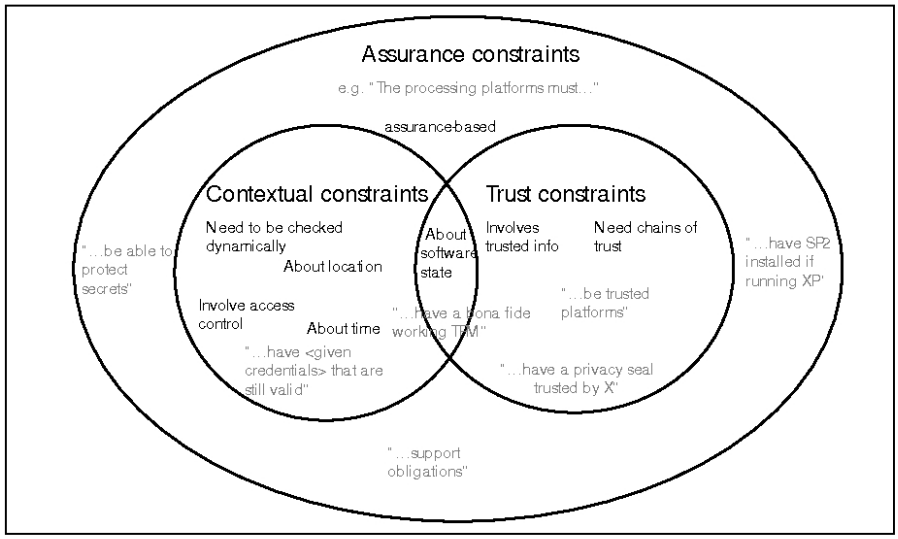


Figure 5: Trust and assurance constraints

Assurance constraints may be contextual constraints, i.e. formulated by people to restrict the cases in which their data will be processed, according to parameters that may vary dynamically (such as

time, location or platform state), or trust constraints. Figure 5 shows how these can be related and provides some examples. To a greater or lesser degree, all assurance constraints could be regarded as trust constraints since something must ultimately be trusted to make the assertions (and indeed the policy compliance checker must be trusted to issue compliance statements), but for convenience we may distinguish those statements that directly involve trust-related information and for whose automated evaluation a compliance checker needs to take chains of trust into account.

These constraints may be expressed within user-side preferences or policies. Such policies (*assurance policies*) would then include a set of conditions and constraints formulated to obtain degrees of assurance from enterprises that their data will be processed according to people's expectations, such as compliance to privacy, security and IT standards. On the client side the Assurance Control module can check their satisfaction (via information provided by the service-side Assurance Control) prior to disclosing any personal information or during the negotiation process (when Assurance Control provides input to both the local user and service-side requester). It can also be desirable to check contextual constraints on the service-side after information has been disclosed. This would be through the use of sticky policies, which are negotiated using the preferences and then associated with data as it travels around, perhaps just using a weak binding, although preferably this would use a strong binding provided by cryptographic mechanisms: see for example [MPB03]. Furthermore, such constraints may be expressed within service-side access control policies to help enterprises comply with privacy legislation, such that service-side Assurance Control will not allow a transaction to be continued unless the constraints are fulfilled.

To clarify exactly what we mean by assurance policies (or constraints), let us consider the W3C Platform for Privacy Preferences (P3P) [CLM+] and Enterprise Privacy Authorisation Language (EPAL) [IBM04] schemas representation of privacy policy rules. These rules are formed of six elements, namely data user, data item, action, purpose, conditions and obligations. Assurance policies could be thought of as an extension to privacy policy rules in that they contain certain trust, contextual or assurance constraints which, if fulfilled, are not sufficient for the transaction to proceed: semantically, these constraints are necessary conditions. An example of an assurance policy which is an access control policy would be:

```
subject with subjexp can action on object with objexp if condition
onlyif assurance constraint.
```

(Alternatively, such an assurance policy could be represented by conjoining the assurance constraint to each subcondition.) There can be other, similar, forms of assurance policy, such as a policy that is attached to data and that contains assurance constraints that must be satisfied before certain actions may be performed on the data.

Within PRIME, we used this approach to define an assurance policy by defining trust-based predicates (including those shown in Figure 5) within the access control policies. Within the PRIME implementation, when such constraints were presented for evaluation to the access control module, the trust-based predicate (i.e. the logical expression involving assurance constraints) is passed to the assurance control module for evaluation, and the result passed to the Access Control Decision component in order to calculate the overall policy satisfaction.

Assurance constraints (including trust constraints) can also be thought of in an orthogonal sense as breaking down into subconstraints, such that there can be functional decomposition of higher-level privacy and trust goals into one or more lower-level goals, and so on recursively until facts about the knowledge base (e.g. checks about the value of constraint settings, the presence of software, the availability of services for a given minimum uptime, etc.) are invoked at the lowest level. This decomposition is captured by rules within our system that hook into the PRIME ontologies used so that the meaning can be agreed across multiple parties. For example, even a fairly low level trust constraint such as that the receiving party should use tamper-resistant hardware to store key information must be defined in such as way as to make clear the manufacturers, version numbers and other ancilliary information such as degree of tamper resistance that would or alternatively would not be acceptable. In practice, a third party would define such rules in advance and then they would be viewable and\or customisable if desired at a later stage by users or administrators.

See [Pea06] for further details about this approach.

## 11.3 Defining Clauses as First Class Objects: A Higher-level Representation

The above representation was used within the initial phases of the PRIME project. However, in the final phases we refined this approach to replace it by making clauses be first class objects. We found

this approach to be preferable to the previous approach, because humans need to read and interpret the assurance policies. Therefore, we wished to tilt the balance in favour of ease of understanding, with the clauses being expressed in natural language, at the expense of the expressivity and richness of the language. We wished to push the complexity of the checking to the third parties involved in the production of evidence, and make things as simple as possible for the end users.

Our model of assurance policies can be analysed by means of different (but equivalent) perspectives, namely the conceptual view, the formal view, and the operational view. We consider these views in turn, and provide examples of assurance polices and the language used.

### 11.3.1 Conceptual View

Both users and service providers have the freedom to create policies to suit their needs. In order to bring the two together a common vocabulary is developed. This comes in the form of privacy statements or privacy clauses which are a basic primitive of our solution. A clause is a statement concerning a particular privacy aspect of PII. It is succinct, clear, and unambiguous and clearly communicates its intended purpose at a level that does not require expert knowledge of privacy systems or their implementation. It is expressed in natural language with the aim that both clients and services will be able to understand each other more clearly. This empowers an end-user, whom it is assumed has no technically advanced knowledge, to communicate their privacy preferences in a language they understand. Later in this chapter we discuss how our policies relate to previous work on policy definition.

A policy is a collection of clauses, crafted for a particular purpose depending on the context of the interaction. Both the user and service provider will invoke the policy that they feel is the most appropriate depending on the context. For the user interacting with a bank they may invoke an "on-line banking" policy; for a service provider interacting with an on-line shopper they may invoke a "website customer" policy. The policies will be geared towards making sense of the context in which they are used. So an "on-line banking" policy may have stricter and more numerous clauses than a "signing up for free email account" policy. It is up to the user and service provider to maintain a pool of polices and invoke them under the proper circumstances. This should then marry up the amount of processing and level of assurance required dependent upon the situation.

There is still an issue about where the clauses come from in the first place, and who provides guidance or establishes what is an appropriate policy for a particular purpose and what is not. In order to facilitate both problems it is important that there be some agreement about privacy in general and clauses and policies in particular. A way of doing this is through standardization. Trusted entities, such as governments or standardization bodies such as the W3C, who have experience in this field through efforts like P3P, can be called upon to provide a working pool of clauses and provide guidance on how to go about creating a privacy policy that is appropriate for a particular activity as a template (see further discussion below).

This approach has an important quality which we have dubbed *privacy positive*. A privacy positive statement is one that is privacy friendly. The clauses are created carefully and worded in such a way to be privacy positive: that is to say that a clause will never reduce the level of privacy afforded to the individual.

The predefinition of clauses is important for three reasons.

1. The clauses are not concerned with technical implementation details: only statements about privacy as required by law, good business practice, and consumer protection will be present. This abstracts away the technical details from the essence of the statements which are only concerned about what should happen with PII and not how it should happen. It prevents restrictions on the way the solutions are implemented and also prevents users from having to be technically savvy to use this scheme.

2. To protect users from having to understand technical details about privacy products and construct detailed policies which may be removed from practical reality, users use clauses that they care about to fashion their policies. In the same vein a service provider, although more technically knowledgeable, uses the same clauses and can speak the same language as its users and can communicate its responsibilities clearly.

3. Since the same pool of statements are being used by both the users and service providers it is an easy matter to match up expected policies with actual ones and negotiate the mismatches. At least

in this way the glaring omissions in service providers' policies will become obvious and in the same way unrealistic expectations from users can be cleared up. Where there are deficiencies in specific clauses, the totality of the policy must be looked at. The set of clauses that form the policy is a stronger indication of the suitability of a policy than the individual clauses of which it is made up. Even if there is disagreement between a user and the service provider at least both know where the other stands on privacy.

We are aware that positive and negative clauses are subjective but it is hoped that through proactive efforts by lawyers and privacy experts in concert with privacy groups it is possible to arrive at a standard of privacy expectations and conduct.

The idea of an *assurance policy template* is an optional extension to this approach, which can be convenient for users in order to help them build up their assurance policies with the help of entities that they trust. An assurance policy template can be thought of a set of default policies (or more specifically, clauses suggested to the user to include within their assurance policy), associated with a given context. For example, an assurance policy template might be suggested by a consumer group for a particular scenario (e.g. purchasing goods of value less than $100 online), and this template would list the checks that the consumer group recommended making in that case. There could be different templates for different contexts, and potentially more than one template for a given context; it would be up to the user to select the appropriate template which they wished to use, if any — this could be done automatically in fact, once the users' initial choices about which templates to use if different preconditions matched were selected and stored.

Templates for policies can provide a set of clauses that adhere to best practices or commonly held standards. To this a user can add or remove clauses depending on their preferences and needs. Templates are especially geared towards end users who may need help creating a privacy policy that would serve the purposes that the end user needed them for.

For example a template for on-line banking may recommend that:

1. PII remains confidential in transit
2. PII is only accessed by authorized personnel
3. A valid privacy seal is present
4. PII is not released to third parties without the consent of the user.

The template can be used in a policy editor to further ease the creation of policies. The end user could use the template as a solid starting point and then tweak it to their desires. This way they can concentrate on their privacy concerns rather than worry about technologies and get distracted from their original intentions. Similarly, a business could also use templates to the same effect although it would have to be careful to only include those clauses it had the actual capability to enforce. A business could not include a clause it could not honour into its policy since the involvement of trusted third parties (TTPs) prevents this type of abuse.

### 11.3.2 Examples of Clauses

Examples of privacy positive clauses can be about any aspect of privacy, from:

- We will not share your data without your consent

to

- We will delete your PII after 30 days

A clause will not break common privacy expectations or allow circumvention by statements such as:

- We will share your data with third parties

or

- We reserve the right to store your data indefinitely

Such privacy negative clauses do not add to the privacy of consumers and would not be valid in privacy policies or adopted in the standard clause pool.

### 11.3.3 Operational view

From a operational perspective, assurance policy templates (aka. compliance checking policy templates) can be seen as collections of compliance clauses. A representation would be:

```
CCPT ctid:
 IF  <pc>
Check L(cc)
```

In a similar way, assurance policies (aka. compliance checking policies) can be seen as collections of clauses as determined by pre-conditions:

```
CCP ctid:
   Check L(cc)
```

For example, *Transfer selected PII if it adheres to Government Policy Template* might correspond to:

```
CCPT ctid1:
IF  templateIS(Government)
CHECK
inEU(ReceivingLocality) AND trusted(ReceivingParty)
```

and *Transfer sensitive selected PII only if it adheres to Secure Storage Policy Template* might correspond to:

```
CCPT ctid2:
IF  templateIS(SecureStorage)
CHECK
NOT(isSensitive(t1)) OR (trustedPlatform(Device) AND
             encrypted(t1,minLevel))
```

## 11.4 Representation of assurance policies in XML format

Within PRIME, we have used an XML format to represent assurance policies. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ccpolicy SYSTEM "ccp.dtd">
<ccpolicy>
<ctid>1337</ctid>
<clause gid="1">
<option>1285</option>
<option>AES</option>
</clause>
<clause gid="2">
<option>128</option>
</clause>
<clause gid="3">
</clause>
</ccpolicy>
```

where `ctid` is a unique identifier to identify each policy, `gid` is a unique well-known global identifier whose mapping to human readable form is standardised, and `option` is a refinement to the clause if applicable.

The corresponding representation of (an example of) the results of the compliance checking process is:

```
<ccpolicy>
<ctid>1337</ctid>
<clause1 gid="1">
<constraint1>128</constraint1>
<constraint2>AES</constraint2>
<result>Y</result>
<signature>abcdef1234567</signature>
</clause1>
<clause2 gid="2">
<constraint1>128</constraint1>
<result>Y</result>
<signature>1341341234124</signature>
```

```
</clause2>
<clause3 gid="3">
<result>Y</result>
<signature>98765787646</signature>
</clause3>
</ccpolicy>
```

The DTD schema underpinning this XML format is:

```
<!ELEMENT ccpolicy (ctid, clause*) >
<!ELEMENT ctid (#PCDATA) >
<!ELEMENT clause (option*) >
<!ELEMENT option (#PCDATA) >
<!ATTLIST clause gid CDATA #REQUIRED >
```

The clauses in the standard clause pool are stored in any suitable form, e.g. a string, or a URI, and each of these is associated with a natural number that is the clause global identifier (`gid`) so that they can be referenced efficiently.

The user assurance policies specify which of these clauses within the standard clause pool the user wishes to check, and the service-side assurance policies specify which clauses within the standard clause pool the service provider is willing to testify that it can provide. These assurance policies are of the same form.

**Processing during a negotiation** As shown in Sec. 10, assurances in PRIME have been integrated into the negotiation process with the goal of having a uniform process for any kind of data exchange. An *Assurance Control* component handles the generation of assurance statements and corresponding evidence by one party and the verification of such statements and evidence by the other party. The Assurance Control component can pull information from the *Trust Management* component of the local platform or multiple platforms comprising a services-side system.

A party $B$, typically service provider, sends their assurance statement ('assurance policy') to the other party in an early phase of an interaction. This statement defines the assurances $B$ promises. This can be done using the part $Rs_{j-1}$ of a message $m_{j-1}$. The same message $m_{j-1}$ contains a request $Rq_{j-1}$ for data to the user as usual. The assurance statements are pulled by the Negotiation component from the Assurance Control component.

At the other party $A$, the Negotiation component passes assurance predicates to the Assurance Control for checking against their own assurance policy that specifies the assurance requirements of the user. Note that this can (at least partially) be performed by the ACD component resulting in a more seamless integration into the architecture. Much like other statements, the assurance statements are displayed to the human user together with the data request. Particularly, the non-fulfilled user requirements are displayed. It is now up to the user, in conjunction with her Assurance Control component and ACD component, to request further details and evidence regarding assurances from $B$. The request parts are composed to the data request $d_j$ that the user anyway issues to the other party.

At $B$, the request for evidence for selected assurance statements is forwarded by the Negotiation component of $B$ to the Assurance Control. The latter returns the requested evidence, either by freshly generating or retrieving it. The evidence is sent back to $A$ using the usual mechanisms for data exchange.

$A$ passes the received evidence elements to its Assurance Control and has them evaluated. Appropriate feedback is given to the human user.

The user can run multiple rounds of requesting assurances from the service provider in order to obtain sufficient information.

The described approach fits almost seamlessly into the negotiation process of PRIME in that assurances are treated homogeneously as data with evidence, much like any other data with a difference that the Assurance Control component handles parts of the processing.

The work on assurances in PRIME has lead to more ideas regarding the processing flow for assurances that may serve useful in practice such as the user being able to provide feedback to the service provider on non-available assurances.

## 11.5 Discussion

During research it was found that assurance ontologies were not the best candidate for assessing the trustworthiness of the back end. The modelling of back end systems was difficult due to problems

of classifying technologies and processes into a coherent assurance ontology that captured all types of systems and variations that are present in real world deployments. Also, this meant that there was a direct link between the technology in use by back-end systems and privacy policies, since privacy polices had to express privacy in terms that back end systems could understand. Another related problem was that even if the model were perfect and complete there existed an expectation that the end user be competent enough to gauge how these technologies benefited them. To avoid these two problems, standardized privacy clauses were introduced, the functionality of which was discussed above.

A policy in the context of this section is the formalization of another party's privacy compliance request. Here, the 'policy' parameter is taken very broadly, and could just include references, or could be a very rich structure. There has been a great deal of work done on privacy policies [CLM+, GH02, KSW02, GK03, BPS03]. In these policy frameworks the focus has been on access control based on conditional logic. Our policies are a departure in that they are not processed against some rule set to produce a decision on whether data should be released. Rather, polices are just collections or groupings of clauses that serve a particular purpose under a particular context. Our solution takes into account that access control plays a big part in the control of PII and so the Assurance Control component works in concert with other components in the PRIME framework, namely the Access Control Decision component (ACD) and the Identity Control component (IDCtrl), to address a variety of aspects needed within a privacy solution, from setting privacy preferences and handling PII requests, to controlling PII release.

P3P is a W3C specification that allows websites and end users to specify their privacy practices and preferences respectively in a standardized way that are easy to retrieve and interpret by end users. It allows a user to delegate the privacy policy "reading" by software agents that compare retrieved website polices against the one created by the user. Only policies that are in violation are flagged to the user who must decide what to do. There have been many critiques of P3P such as [Clab, Ack04, GH02, Claa]. We shall ignore politico-economic arguments and focus on how our solution differs from P3P, the gaps it fills in, and how P3P could be used within the system we have implemented albeit with changes to its role.

Expressing privacy concerns in P3P is done by defining statements in a machine readable format written in XML [CLM+]. Although there are editors [P3P] that help with this process, there are two problems that are not yet addressed.

First, the P3P language and editor are tools but the end user must know what they wish to express in the first place. They must know what their privacy vulnerabilities are and how to check if a website will mitigate those risks. Most users are na¨ve and would not be competent enough to express privacy concerns beyond vague statements.

Second, even with the prerequisite privacy knowledge the definition of privacy polices must be in a language geared towards the facilitation of accessing PII based on conditions. Although useful it cannot capture other aspects of privacy adequately without losing some of the essence of what the end user intended. Our solution addresses these concerns by introducing standardised privacy clauses that are written in human readable form and are unambiguous, concise, and capture privacy concerns based on expert knowledge. To ease the creation of polices, templates are provided. The end user does not need to learn a language or an editor that requires knowledge of predicate logic.

As is the case with privacy seals, P3P cannot link the privacy practices expressed by the website with anything tangible on the back-end. This gap is where our solution introduces mechanisms to check that policies and the technical realities of the website's infrastructure are coherent. Claims made in the privacy policies are backed up by capability checks and help, e.g,. carried out in a negotiation, to provide assurances that are missing from the P3P model.

Although P3P has its limitations, its strength as a robust policy definition language and logic model allows it to perfectly translate privacy clauses into machine readable form. The resultant privacy policy would have to be vetted by TTPs and also certified, or an intermediate layer could be introduced that would drive the policy editor to receive clauses and output machine readable policies. Since the clauses are defined and standardised the resultant XML would also be identical. In our model unique global identifiers are used to identify particular clauses, the drawback being that a unique identifier needs a lookup table to be maintained, whereas an XML policy would capture all the necessary information within itself. There have to be extensions to the present P3P vocabulary so that all aspects of privacy can be expressed.

### 11.6 Next steps and future R&D work

A policy in the context of this section is the formalization of another party's privacy compliance request. Here, the 'policy' parameter is taken very broadly, and could just include references, or could be a very rich structure.

We have used a common standardized privacy clause pool to help communicate end user concerns as well as service provider promises. These clauses form high level assurance checking policies. The benefits of this approach are in providing flexibility, and in being extensible and customisable.

Since trust is not a black and white issue, we designed this approach such that the user must have overall control over how to proceed. Nevertheless, there is subjectivity and potential changeability of the decisions and representations involved, which could be an issue.

Since clauses are the central privacy vector they need to be developed further from the select set that are being implemented now. They need to be more complex and recognise complex privacy needs of sophisticated users as well as laws and regulations that businesses must adhere to. They also need to be stated in such a way that is unambiguous in any language. Only the true essence of the privacy objective of the clause must be present in its description. This will be an interesting area which will require participation from law, business, and security experts for further refining and establishing a coherent, effective, and simple language for defining privacy issues and concerns.

Further aspects about the integration of assurance control functionality are given in Sec. 10.

### 11.7 Assurance Control Component

Note that many of the mechanisms for assurances can be circumvented by an arbitrarily dishonest party on standard data processing systems. Still, the approach is very useful in order to give honest service providers a tool that allows users to gain confidence in the data processing practices of the service provider. When assurance mechanisms will be based on upcoming integrity-protected computing technology, the method will work in a stronger trust model and make the overall approach even more appealing.

The system complexity of the system for gathering measurements and performing tests can be arbitrarily complex. This complexity is hidden by having the technicalities abstracted through the assurance statements that are used by other components and communicated between parties.

## 12 Privacy-aware Identity Lifecycle Management: Principles and concepts

So far, we have extensively discussed the release of data between parties and how this can be performed in a privacy-enhancing manner, culminating in our presentation of the PRIME negotiation protocol in Sec. 10. This section is dedicated to the discussion of the aspects of privacy enforcement after data have been released. The approach we discuss here is the necessary complement to the negotiation protocol: In a negotiation protocol, privacy obligations are agreed on for the data to be released, and those privacy obligations need appropriate enforcement once the data enter the recipient's domain of control. This is subject of the elaborations below.

### 12.1 Introduction

Privacy-aware identity lifecycle management processes must be put in place by enterprises to effectively manage the lifecycle of personal and confidential information according to privacy (law) requirements - over time and across various contexts and solutions. As anticipated, this includes dealing with data retention, data deletion, satisfying notice requirements, supporting data transformations and management of complex workflows. Privacy obligation policies can be used to express these expectations and also keep into account user preferences.

This requires a well-planned, systemic and ongoing effort, because: privacy obligation policies and personal preferences can change over time; data and confidential documents can be subject to different privacy and data protection laws depending on geographical and organisational boundaries; data needs to be disposed or transformed over time. The lifecycle of the involved privacy policies must be managed as well.

## 12.2 Obligation Management Framework

As anticipated in the "Privacy Models and Languages" Chapter, an Obligation Management Framework is introduced to explicitly handle privacy obligations.

In our vision, at the core of privacy-aware identity lifecycle management solutions there is an Obligation Management Framework to centralise (within enterprises) the representation and management of privacy obligations and orchestrate their overall enforcement and monitoring by leveraging and extending current enterprises IT solutions, in particular Identity Management solutions.

Figure 6 shows the conceptual model underpinning this framework.

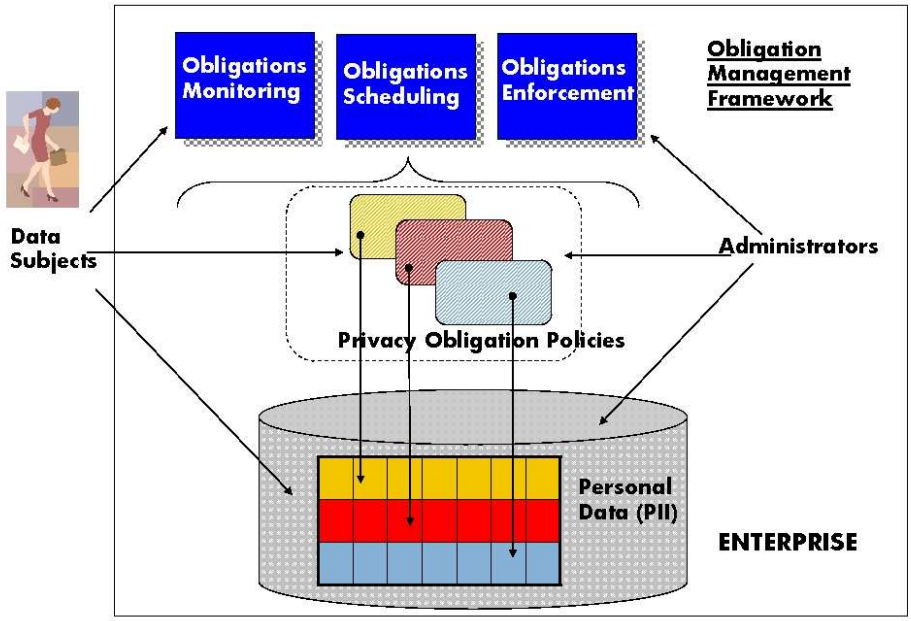


Figure 6: Proposed Privacy Obligation Management Model

In our model privacy obligations are independent entities that are explicitly modeled and managed to enable a privacy-aware lifecycle management of personal data. They are not subordinated to access control aspects. Data subjects can define privacy obligations and associate them to their personal data at the disclosure time (e.g. during a self-registration process) or at any subsequent time. Enterprise privacy administrators can also associate additional privacy obligations, for example dictated by laws or internal guidelines. In our model, the obligation management framework handles these obligations and their associations to personal data by providing the following core functionalities:

- Explicit modeling and representation of privacy obligations: a language/format is defined to explicitly represent privacy obligations in order to analyse them and reason about their implications;
- Scheduling the enforcement of privacy obligations: the system schedules which obligations need to be fulfilled and under which circumstances (events);
- Enforcing privacy obligations: the system enforces privacy obligations once they are triggered. The enforcement ranges from the execution of simple actions to complex workflow involving human interventions;
- Monitoring the fulfillment of privacy obligations: the system monitors and audits the enforced obligations, at least for a predefined period of time, to ensure that the desired status of data is not violated and to report anomalies;

- Administration and lifecycle management of privacy obligations.

These functionalities can be accessed by enterprise privacy administrators and potentially by data subjects, for example to monitor their personal data and check for privacy compliance.

## 12.3 Obligation Management System

At the very core of this obligation management framework there is an Obligation Management System, in charge of dealing with the enforcement and monitoring of privacy obligations and interacting with other components, such as the privacy-aware access control component.

### 12.3.1 Design Rationale

The design rationale behind our obligation management system is dictated by the requirements and issues described in the "Privacy Models and Languages" Chapter and based on our privacy obligation model and obligation management framework.

As previously anticipated, in our approach privacy obligations are handled in an explicit way, independently and not subordinated to access control. This is required in order to deal with privacy obligations that involve deletion of data, notifications or complex workflows, requests for authorizations and executions of workflows that must be triggered independently by access control activities.

Based on this, our design choices reflect the following core aspects:

1. Privacy obligations are self-standing policies, represented with an appropriate language, separated from access control policies;

2. The obligation management system must explicitly parse, manage, schedule, enforce and monitor privacy obligations via dedicated modules. In particular, the monitoring of enforced obligations is important to ensure that the overall system is compliant to enforced privacy obligations and that violations are spotted and reported to administrators.

The fact that the obligation management system must handle privacy obligations over long-periods of time and must be always available has also influenced our design choices: survivability and reliability are core requirements. The current design of the obligation management system takes these requirements into account: it is possible to create multiple distributed instances of the obligation management system and monitor for their availability.

### 12.3.2 System Architecture

Figure 7 shows a high-level architecture of an obligation management system supporting the explicit management and enforcement of privacy obligations.

This obligation management system consists of the following modules:

- **Obligation Server**: it deals with the authoring, management and storage of obligations. It explicitly manages the association of privacy obligations to confidential data and their tracking and versioning. It pushes active obligations (i.e. obligations to be fulfilled) to the "obligation scheduler". One or more obligation servers can be deployed (and synchronised), depending on needs;

- **Obligation Store and Versioning**: it stores obligations and their mapping to confidential data. Multiple versions of obligations can also be stored in this system, though in the current version of the system this functionality has not yet been implemented;

- **Obligation Scheduler**: it is the module that knows which obligations are active, ongoing obligation deadlines, relevant events and their association to obligations. When events/conditions trigger the fulfillment of one or more obligations, this component activates the correspondent "workflow processes" of the "obligation enforcer" that will deal with the enforcement of the obligation;

- **Obligation Enforcer**: it is a workflow system containing workflow processes describing how to enforce one or more obligations. The enforcement can be automatic and/or could require human intervention, depending on the nature of the obligation. It is extensible via plug-ins, each of them providing a specific enforcement functionality;

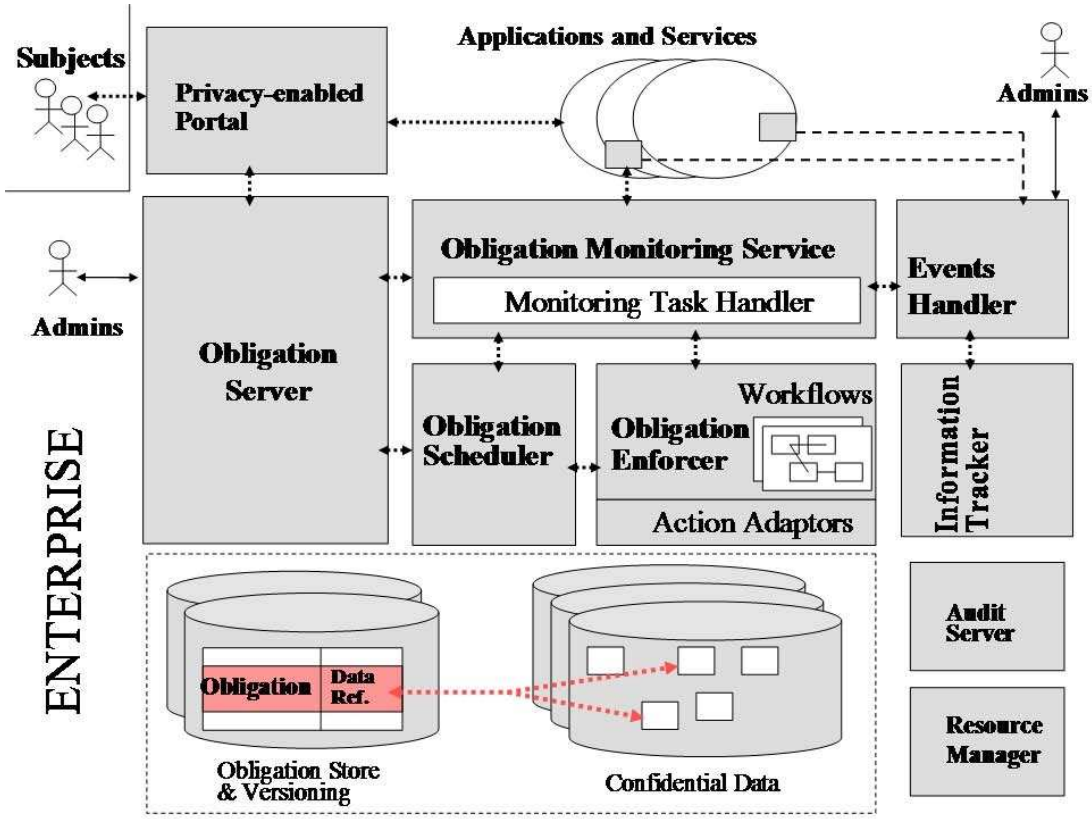


Figure 7: High-level Architecture

- **Events Handler**: it is the module in charge of monitoring and detecting relevant events for privacy obligations and sending them to the obligation scheduler. The detection of events can happen via instrumented application/services. They can also be directly generated by users, administrators, the "obligation monitoring service" and the information tracker;
- **Obligation Monitoring Service**: it is the module, orthogonal to the scheduling and en-forcement systems, that monitors enforced obligations by analysing and checking for the effects of their actions i.e. if the personal data targeted by the obligation is in the desired state;
- **Information tracker**: it is a module that focuses on intercepting events generated by data repositories, databases and file systems containing confidential data and providing this information to the event handler. It is aware of the location of confidential data (as described by the obligation policies) and checks for movements and changes happening to this data;
- **Audit Server**: it audits the relevant events and information generated by the overall system modules and involved applications/services;
- **Resource Manager**: it is a module in charge of checking that all the other system components are running and allocating their services to requestors.

The core "run-time" functionalities provided by a system based on this architecture include:

- **Setting a new privacy obligation**, Figure 8: a new obligation is sent to the Obligation Server, either by a data subject or an administrator. The Obligation Server parses and checks for its format correctness. It stores this obligation in the obligation database and communicates it to the Obligation Scheduler to ensure that the obligation will be processed at the due time;
- **Enforcing a privacy obligation**, Figure 9: the Obligation Scheduler listens to managed events sent by the Event Handler and checks if any of them (or any combination of them) triggers one of the managed obligations. Should this happen, the Obligation Scheduler communicates with the

Obligation Server to retrieve all the relevant information and sends the obligation to the Obligation Enforcer. The Obligation Enforcer analyses the "action part" of the obligation and executes all the listed actions. Independently by the enforcement result, it sends a copy of the obligation to the Obligation Monitoring Service;

- **Monitoring an enforced privacy obligation**, Figure 10: the Obligation Monitoring Service periodically checks the status of personal data, against related privacy obligations that have been enforced. This is important for compliance reasons, to identify possible violations or technical problems. For example, in case of deleted data (as a consequence of enforcing an obligation) this module will check if data are actually deleted, for a predefined period of time. It might happen that, because of wrong database synchronisation or back-ups, deleted data reappears in the repository: our system will be able to spot this anomaly.

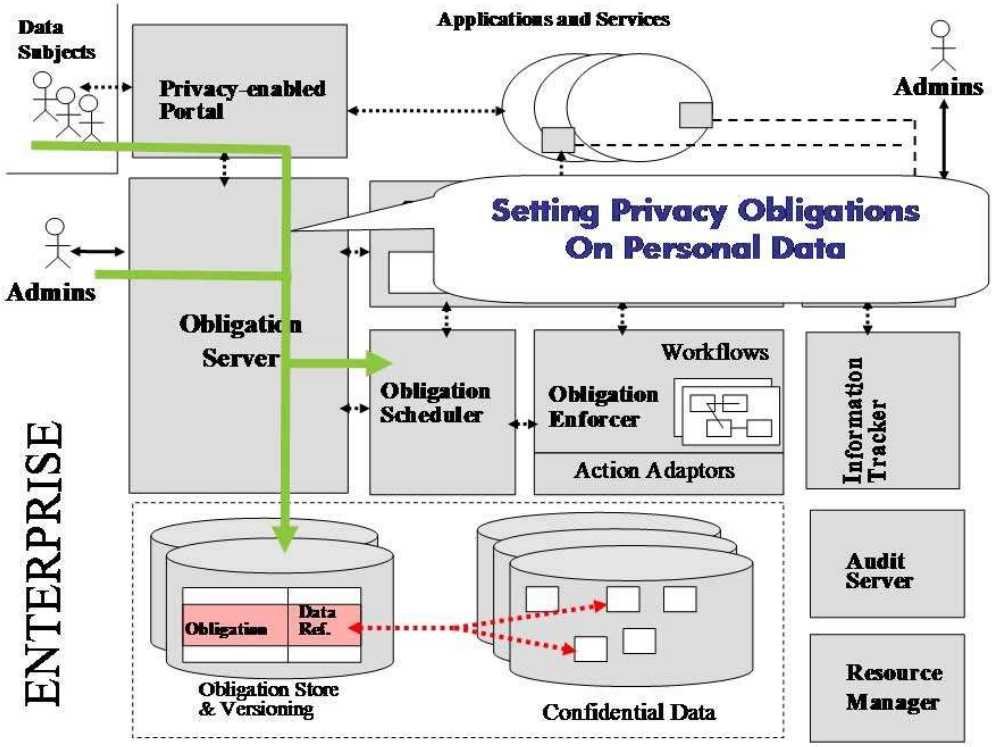


Figure 8: Setting a New Privacy Obligation

The privacy obligation management system is a critical system: it must survive faults and excessive workloads.

Our system has been built to be distributed and the instantiations of its components can be replicated.

Multiple distributed instances of all the above components can be created and run in parallel: all of them are stateless, as the relevant information on managed privacy obligations is stored in a replicated database. A (replicated) Resource Manager module manages these instances and allocates these resources to requesters (for example the Obligation Server trying to connect to an Obligation Scheduler or the Obligation Scheduler trying to connect to an Obligation Enforcer).

## 12.4 Operational View of Privacy Obligations

The discussions we have next, complement the conceptual view of privacy obligations as presented in Sec. 9.2 with their operational perspective. From an operational perspective, privacy obligations can be seen as reactive rules [RHMP05], i.e., rules that are triggered by events and/or by the fact that the specified conditions are met. As an effect (reaction) of triggering a rule, actions are executed.

A representation of privacy obligations as reactive rules follows:

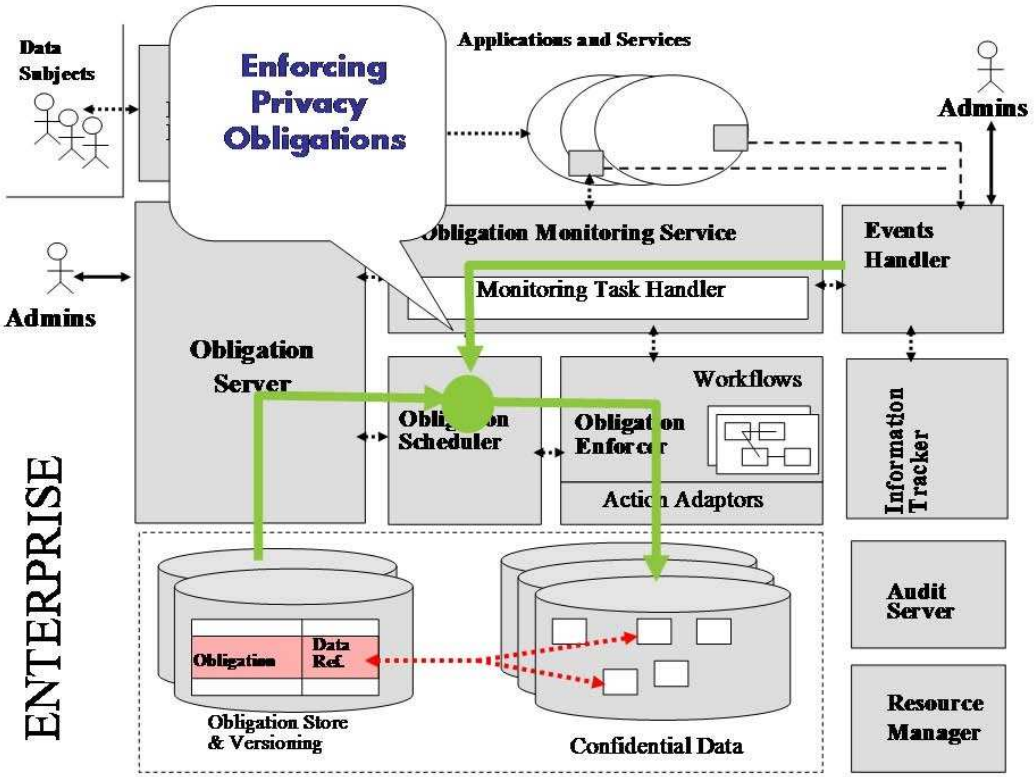


Figure 9: Enforcing a Privacy Obligation

```
OBLIGATION Oid:
   TARGETS: t
   WHEN L(e)
   EXECUTE C(a)
```

In this context, given an obligation with unique identifier Oid and a target t, if the logical combination of events L(e) is true, i.e. it triggers the rule, then the combination of actions C(a) has to be executed. The remaining part of this thesis will focus on this operational definition of privacy obligations.

As anticipated at the beginning of this section, privacy obligations are associated to personal data and can be defined by data subjects and privacy administrators. A few simple examples of privacy obligations follow:

```
OBLIGATION Oid1:
TARGETS:
  t1:< DATABASE=db1, TABLE=customers, Key=CustomerName, KeyValue=abc>
  WHEN (current_time= date1)
  EXECUTE <DELETE t1>
```

In this example, a customer record, stored in a specified table of a database, must be deleted at a well defined point of time. This is a simple example of a data deletion obligation.

```
OBLIGATION Oid2:
TARGETS:
  t1:< DATABASE=db1, TABLE=customers, Key=CustomerName, KeyValue=abc,
  ATTRIBUTES=(e-mail) >
WHEN (Access_Data_Event AND Access_Data_Event.data = t1)
EXECUTE  <NOTIFY BY t1.e-mail>
```

In this example, when an event (for example issued by an access control system) indicates that a specific customer's record has been accessed, a notification has to be sent to the customer, by using his/her e-mail address.

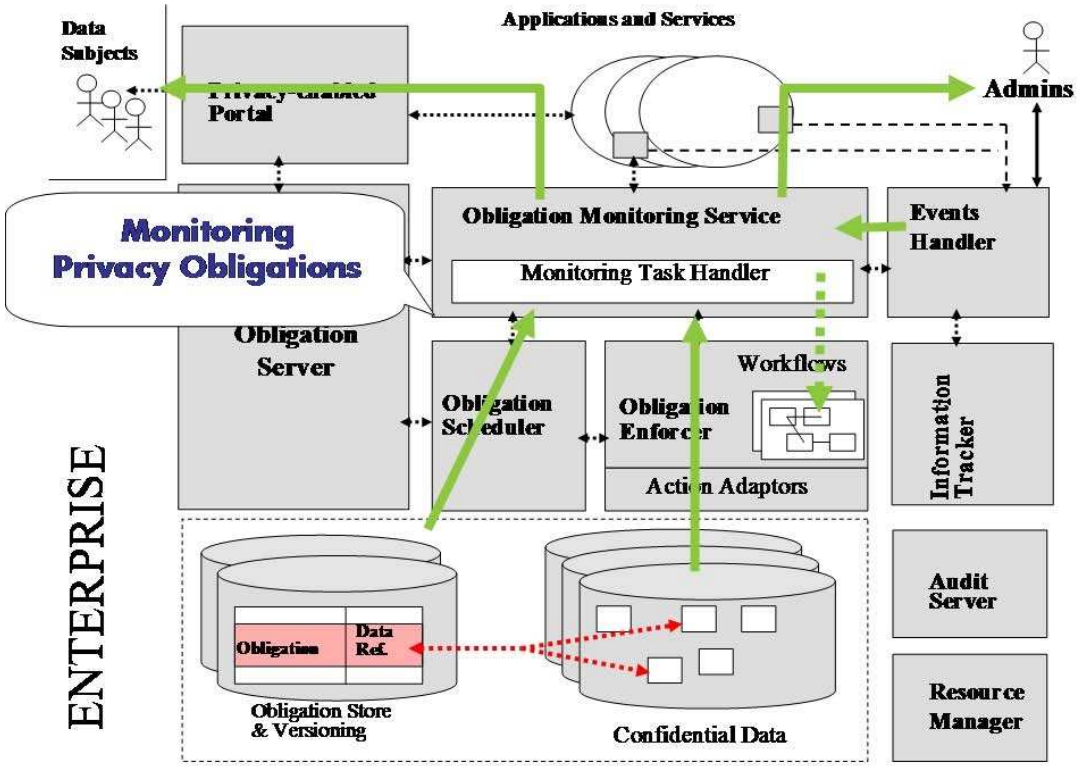


Figure 10: Monitoring a Privacy Obligation

```
 OBLIGATION Oid3:
 TARGETS:
  t1:< DATABASE=db1, TABLE=customers, Key=CustomerName, KeyValue=abc
ATTRIBUTES=(creditcard,e-mail)>
WHEN
     (current_time>date1)
        AND
    (NOT (Access_Data_Event AND Access_Data_Event.data = t1 ))
EXECUTE
   <NOTIFY BY t1.e-mail>
   <DELETE t1.creditcard>
```

In this example, if customer's data is not accessed after a predefined amount of time, an attribute (credit card) has to be deleted and the customer must be notified.

```
 OBLIGATION Oid4:
 TARGETS:
   t1:< DATABASE=db1, TABLE=customers, Key=CustomerName, KeyValue=abc
   ATTRIBUTES=(creditcard,e-mail)>
WHEN
     (current_time>date1)
        OR
        ( (Access_Data_Event AND Access_Data_Event.data = t1 )
            AND
          (Access_Counter>n))
EXECUTE
   <DELETE t1.creditcard>
   <RUN WORKFLOW deprovision_user(t1.KeyValue)>
```

In this example customer's data is deleted and the customer account is de-provisioned (accounts deleted, access rights revoked, etc.) from various IT systems either at a specific point in time or after customer's data has been accessed more than n times.

```
 OBLIGATION Oid5:
```

```
 TARGETS:
  t1:< DATABASE=db1, TABLE=customers, Key=CustomerName, KeyValue=abc,
  ATTRIBUTES=(e-mail)>
WHEN
  (current_time < date1)
    AND
  (time_counter > time_interval)
EXECUTE
 <NOTIFY BY t1.email>
 <RESET time_counter>
```

In this example, periodic (ongoing) notifications are sent by e-mails to customers, for example to notify them about the fact that the enterprise is retaining their personal data. This is an ex-ample of ongoing obligation. Specific types of privacy obligations can be set-up by enterprise privacy administrators to handle personal data based on internal guidelines and/or laws. These privacy obligations can be triggered by internal events determined by contextual and infrastructural changes. A few examples follow.

```
 OBLIGATION Oid6:
 TARGETS:
  t1:< DATABASE=db1, TABLE=customers>
WHEN
     (Event-intrusion_detected)
EXECUTE
   <ENCRYPT  t1>
   <NOTIFY admin>
```

In this example, we assume that an intrusion detection system is able to send alerts to sub-scribers (including our obligation management system) when intrusion attempts are detected. A privacy obligation can be triggered to protect the entire content of a "confidential" table by encrypting its content and notifying the administrator. This action can be seen as a best-effort, temporary solution to prevent that personal data are accessed by the intruder.

```
 OBLIGATION Oid7:
 TARGETS:
  t1:< DATABASE=db1, TABLE=customers>
WHEN
     (Event-system_distrusted)
      AND
      (DATABASE.host =system_distrusted.host)
EXECUTE
   <ENCRYPT  t1>
   <NOTIFY admin>
```

Similarly to the previous obligation, this obligation is triggered by contextual changes. In this case, one of the systems hosting the database is classified as "distrusted" (because of a virus infection, locally detected spyware, installation of dubious software, etc.) by enterprise moni-toring systems. If this event is sent to the obligation management system, this system can trig-ger the above obligation that will encrypt the data and notify the administrator. Again, this ac-tion can be seen as a best-effort, temporary solution to prevent that personal data is compromised.

```
 OBLIGATION Oid8:
 TARGETS:
    t1:< FILE=../audit_log, ATTRIBUTES=(TimeStamp, UserIPaddress, UserName)>
WHEN
  (time_counter > time_interval)
EXECUTE
 <ENCRYPT  t1.UserIpAddress>
<DELETE t1.UserName WHERE t1.TimeStamp<= current_time - 6 months>
<RESET time_counter>
```

This privacy obligation is defined by a privacy administrator to "purge" the content of an audit log file (for example created by a web server) of specific personal data, as dictated by internal guidelines (for example after six months) and encrypt another portion of the data that can be decrypted later on, in case of need. This is an example of ongoing obligation that is periodi-cally triggered based on a predefined interval of time (for example every week).

All the actions described in the above examples of privacy obligations can (conceptually) be generalised as workflows.

A workflow consists of one or more actions/tasks to be executed, in a specified order. In the remaining part of this thesis the concept of workflow is implied whenever privacy obligation's "actions" are discussed. Please notice that the language used in the above examples to describe privacy obligations is purely illustrative.

## 12.5 Discussion and Conclusions

The management of privacy is very important for enterprises in order to deal with regulatory compliance and customer satisfactions aspects. In particular privacy obligations need to be managed. In this context solutions are required to automated privacy-aware information lifecycle management and reduce costs. Their scalability to large set of data is a key requirement.

In this chapter we described work done in PRIME to explicitly represent, enforce and monitor obligation policies within organisations. A description of an Obligation Management System has been provided in full details, demonstrating how to achieve this.

To address important scalability issues, extensions that are not described here have been developed. More precisely, better scalability is achieved by factoring in parametric obligation policies and a scalable obligation management system and framework.

A working prototype has been fully implemented and integrated with HP identity management solutions to show the feasibility of our approach in a real world domain. Initial tests demonstrate the scalability of our approach to handle obligation policies on large sets of data (more than 100K records).

# 13 Conclusions

During the process of developing the PRIME Architecture, the team involved in the work has gained *substantial experience* in the area of integrating multiple PETs with the goal of orchestrating them such that our goals towards privacy protection are reached. We have experienced a rather high complexity on the technological side in our architectural efforts, but could master this complexity and have succeeded in the integration efforts with the result being a comprehensive architecture for privacy-enhancing identity management that follows the user centric paradigm. Note that our results are mainly at the level of common infrastructure for identity management that can be applied independently from specific application domains.

Our component architecture in its latest version is, compared to the earlier version, sufficiently *modular* such that individual components can be deployed separately. This is a precondition towards deployment of our technology in practice as this helps to reduce complexity and solve exactly one specific problem. Additionally, this componentization reduces complexity to a level that can be communicated to potential stakeholders. It was a substantial effort to get to the current modularization. The *data model* has been identified as being crucial in the effort of proper componentization. One of our key lessons learnt is to not underestimate the importance of the data model.

One of the key goals that we have reached at the technical level is the *reduction of required trust* in parties such as service providers and certifiers, particularly by the user of technologies like anonymous credential systems for data exchange. This leads to a stronger system in terms of data protection for a user. Compared to today's situation, a user's privacy can even be protected if service providers and certifiers are dishonest, if business processes are appropriately defined and PRIME technology is used. This is a major advancement in terms of what can be done for the user, but we still face major obstacles towards a deployment of such technology in the field at a large scale. In our view, the part of convincing businesses to design their business processes in a way such that data minimization can be implemented as envisioned in PRIME will even be harder than has been the technological part. We made the observation that legal compliance is the key goal for businesses, everything going beyond this is harder to motivate unless it directly results in profit or otherwise added value for the business. Still, we think that many arguments speak for privacy and those need to be properly assessed by businesses in making their decisions and that an attitude towards better privacy is (slowly) emerging on the business side as well in Europe. For more traditional scenarios where a user must still provide identifying personal data to a service provider, we have built tools that allow a user to better judge the trustworthiness of a service provider by performing an *assurance assessment* of the service provider.

Another key goal that has been reached in developing technologies that do not help to reduce the trust requirements in parties, but that result in tools that can help enforce privacy better by *automating enforcement of policies* at various stages. Our approach to the life cycle management of data with the automated enforcement of data handling policies is a good example here. Specific mention here deserve the *privacy obligation management* system of PRIME that has been tested with large data sets.

On the requirements side, we have experienced a very *diverse landscape of requirements* in different domains and at different levels for a system like PRIME. In the early phases of the project it was a challenge to sort out the key requirements that should drive our technology development. Over time, it became quite clear what the key requirements are and our work was driven to a large extent by those. Particularly difficult were requirements at the social and economic level as they can mostly not be directly reflected in technology, but rather require a concerted approach in various disciplines.

The tradition of *user centricity* and privacy in identity management of PRIME will be followed up on by the Project PrimeLife. This project is tackling more specific areas and, in contrast to PRIME, targets collaborative scenarios in the paradigm of user-generated content more specifically. The PrimeLife project will, as appropriate, leverage the infrastructural results of PRIME and carry them over to new classes of applications.

We want to note that during the course of the PRIME project, the space of user-centric identity management has gained substantial traction and has been moving very fast. There is no end yet in this development and as of today, the space has shifted more towards privacy as well. Even advanced technologies such as anonymous credential systems are under evaluation by major players in the field.

Considering the above discussions, it can be concluded that the insights we have gained in the space of privacy-enhancing identity management have been worth the effort taken. We are confident that some of our results and ideas will be picked up by others and reach the citizen in some way in the mid term. Overall, when looking at the development of the identity management area over the lifetime of PRIME,

one can be very optimistic about the future of privacy-enhancing identity management.

# References

[ACDS07] C.A. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati. A privacy-aware access control system. *Journal of Computer Security (JCS)*, 2007. (to appear).

[Ack04] M. S. Ackerman. Privacy in pervasive environments: next generation labelling protocols. In *Personal Ubiquitous Computing 2004*, pages 430–439. Springer-Verlag, 2004.

[Act03] Gramm-Leach-Bliley Act. Gramm-leach-bliley act: Privacy of consumer financial information. 2003. `http://www.ftc.gov/privacy/glbact/glboutline.htm`.

[ADS06] C.A. Ardagna, S. De Capitani di Vimercati, and P. Samarati. Enhancing user privacy through data handling policies. In *Proc. of the 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, SAP Labs, Sophia Antipolis, France, July-August 2006.

[AHK+03] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise privacy authorization language (epal 1.1). 2003. `http://www.zurich.ibm.com/security/enterprise-privacy/epal`.

[BPS03] M. Backes, B. Pfitzmann, and M. Schunter. A toolkit for managing enterprise privacy policies. In E. Snekkenes and D. Gollmann, editors, *ESORICS 2003*, LNCS 2808, pages 162–180. Springer-Verlag, October 2003.

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

[BSCGS06] Abhilasha Bhargav-Spantzel, Jan Camenisch, Thomas Gross, and Dieter Sommer. User centricity: a taxonomy and open issues. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management*, pages 1–10, New York, NY, USA, 2006. ACM.

[Buc04] Prime Requirements Version 0 – Part 1: Legal Requirements, October 2004. `http://www.prime-project.eu`.

[Buc05] Anna Buchta, editor. *PRIME Requirements Version 1 – Part 1: Application Requirements.* May 2005. `http://www.prime-project.eu`.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, 2001.

[CL02] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer, 2002.

[Claa] R. Clarke. P3p re-visited. `http://www.anu.edu.au/people/Roger.Clarke/DV/P3PRev.html`.

[Clab] R. Clarke. Platform for privacy preferences: A critique. `http://www.anu.edu.au/people/Roger.Clarke/DV/P3PCrit.html`.

[CLM+] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification.* `http://www.w3.org/TR/P3P/`.

[CMP07] Stephen Crane, Marco Cassasa Mont, and Siani Pearson. On helping individuals to manage privacy and trust. In *Trust Management*. Icfai University Press, June 2007.

[CPR+08] Jan Camenisch, Ulrich Pinsdorf, Thomas Roessler, Rigo Wenning, Kai Rannenberg, Pierangela Samarati, Mario Verdicchio, Markulf Kohlweiss, Stuart Short, Karel Wouters, Carine Bournez, Eduard de Jong, Robert Mueller, Michele Bezzi, Immanuel Scholz, Claudio d'Ardagna, Benjamin Kellermann, and Patrik Bichsel. First report on standardisation and interoperability and overview and analysis of open source initiatives. deliverable, PrimeLife, May 2008.

[Eur95] European Parliament. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal L*, pages 31–50, 1995.

[Eur02] European Parliament. Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of persomal data and the protection of privacy in the electronic communications sector. *Official Journal L*, pages 37–47, 2002.

[eXt05] *eXtensible Access Control Markup Language (XACML) Version 2.0*, February 2005.

[FHA04] Simone Fischer-Hübner and Christer Andersson, editors. *Framework V0*. 2004. `http://www.prime-project.eu`.

[FHAH05] Simone Fischer-Hübner, Christer Andersson, and Thijs Holleboom, editors. *Framework V1*. 2005. `http://www.prime-project.eu`.

[FHH08] Simone Fischer-Hübner and Hans Hedbom, editors. *Framework V3*. 2008. `http://www.prime-project.eu.org`.

[GH02] M. Wilikens G. Hogben, T. Jackson. A fully compliant research implementation of the p3p standard for privacy protection: Experiences and recommendations. In D. Gollmann et al., editor, *ESCORICS 2002*, volume 2502 of *LNCS*, pages 104–125. Springer-Verlag, 2002.

[GK03] M. Waidner G. Karjoth, M. Schunter. Platform for enterprise privacy practices: Privacy-enabled management of customer data. In R. Dingledine and P. Syverson, editors, *PET 2002*, volume 2482 of *LNCS*, pages 69–84. Springer-Verlag, 2003.

[Hig] Higgins project. `http://www.eclipse.org/higgins/`.

[HS06] Giles Hogben and Dieter Sommer. A meta-data and reasoning framework for open assertion and evidence exchange and query. Technical report, IBM Research, October 2006.

[IBM04] IBM. The enterprise privacy authorization language (epal). `http://www.zurich.ibm.com/security/enterprise-privacy/epal/`, 2004.

[KN03] Chris Kaler and Anthony Nadalin. Web services federation language, 2003.

[KSW02] G. Karjoth, M. Schunter, and M. Waidner. Privacy-enabled services for enterprise. In *DEXA'02*. IEEE, 2002.

[LFH06] Ronald Leenes and Simone Fischer-Hübner, editors. *Framework V2*. 2006. `http://www.prime-project.eu`.

[lib] Liberty Alliance. `http://www.projectliberty.org/`.

[MCKS06] Marco Casassa Mont, Stefano Crosta, Thomas Kriegelstein, and Dieter Sommer, editors. *PRIME Architecture Version 2*. March 2006. `http://www.prime-project.eu`.

[Mic05] Microsoft. A technical reference for InfoCard v1.0 in windows, 2005.

[MPB03] M. Casassa Mont, S. Pearson, and P. Bramhall. Towards accountable management of privacy and identity management. In *Proc. ESORICS*, 2003.

[Ope] Openid. `http://openid.net/`.

[P3P] P3P. Implementation report. `http://www.w3.org/P3P/implementation-report.html`.

[Pea06] S. Pearson. Towards automated evaluation of trust constraints. In K. Stlen et al., editor, *Proc. iTrust 2006*, volume 3986 of *Lecture Notes in Computer Science*, pages 252–266. Springer Verlag Berlin Heidelberg, May 2006.

[PHKS02] P.Ashley, S. Hada, G. Karjoth, and M. Schunter. E-P3P privacy policies and privacy authorization. In *Proc. of the ACM workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.

[RHMP05] T. Roessler, G. Hogben, M. Casassa Mont, and S. Pearson. Position paper: Rule language requirements for privacy-enabled identity management. 2005. `http://www.w3.org/2004/12/rules-ws/paper/59/`.

[Sch08] Günter E. Schumacher, editor. *PRIME Requirements Version 3.* 2008. To appear on `http://www.prime-project.eu`.

[Som04] Dieter Sommer, editor. *PRIME Architecture Version 0.* October 2004. `http://www.prime-project.eu`.

[Som05] Dieter Sommer, editor. *PRIME Architecture Version 1.* August 2005. `http://www.prime-project.eu`.

[W3C04] W3C. Resource description framework (rdf). 2004. `http://www.w3.org/RDF/`.

[Wil04] Marc Wilikens, editor. *PRIME Requirements Version 0 – Part 3: Application Requirements.* October 2004. `http://www.prime-project.eu`.