MASTER

Access policy enforcement techniques to protect data shared across domains

Pletea, D.

Award date:
2014

Technische Universiteit Eindhoven University of Technology
Department of Mathematics and Computer Science
Security Group

Access Policy Enforcement Techniques to Protect Data Shared Across Domains

Daniel Pletea

Graduation Supervisor:	
	prof. dr. Milan Petkovic
Daily Supervisors:	
	ir. Paul Koster
	dr. Saeed Sedghi
External Committee Member:	
	dr. Rudolf Mak

Eindhoven, August 2014

Abstract

Data sharing across domains is nowadays becoming more popular and common. Such collaborations are able to open valuable new research opportunities, while lowering the costs of access to data. These benefits come with increased number of security concerns regarding authorizing access to data across domains. These concerns translate into security requirements and later into access policies that have to be enforced by all the collaboration parties. This can be difficult as the authorization techniques that the collaborative parties are using might be different. Existing cross-domain authorization mechanisms present shortcomings with regard to efficiency, access policies conflicts solving and granularity of the access control.

In this work we propose an efficient solution for cross-domain policies administration and enforcement. We design a two-layer authorization architecture to enforce access policies, where the first layer uses shared collaboration policies. This layer is overruling the second layer which consists of domain-specific access policies. The proposed solution is evaluated and proves to be more efficient than the traditional approach. Additionally, we design a solution for dealing with data sensitivity status using data sensitivity annotations. We develop a mechanism which uses the annotations to mask access to data. Next we validate that this solution is masking data efficiently and is capable of providing fine-grained access control. Moreover the annotations are making use of user geolocation. We prove that the data masking process is enforcing user geolocation successfully in a proof of concept application service.

Acknowledgments

This thesis report is the result of my graduation project, which concludes my master program Computer Science and Engineering, at the Eindhoven University of Technology. The project was carried out within the Data Science department of Philips Research B.V. in collaboration with the Security group (SEC) of the Department of Mathematics and Computer Science at the Eindhoven University of Technology.

I would like to express my deepest gratitude to my supervisor, Prof. Dr. Milan Petkovic, who gave me the great opportunity of choosing this project and for his careful guidance throughout this period. I am grateful for his suggestions and knowledge, which were invaluable for the completion of this work.

My profound gratitude also goes to my daily supervisors, Ir. Paul Koster and Dr. Saeed Sedghi, who guided me during the whole process, helped me with the encountered research challenges and assisted me in writing the report. Their comments, suggestions, and advice helped me in solving the problems addressed in this thesis. The several constructive discussions I had with them were invaluably helpful, gave me sound advice and a lot of new insights into the researched topic.

Furthermore, I would like to thank all my colleagues at Philips for providing a propitious and stimulating environment in which to learn and progress.

Last but not least, I would like to thank my parents and all my friends for their support and encouragement during the whole experience.

Daniel Pletea
Eindhoven, August 2014

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Description

Nowadays collaborations and delivery of services across different organizational domains are gaining popularity. In such systems, usually data is shared between distinct systems and accessed from different global locations, using a cloud computing or any other appropriate platform. By sharing of data among the collaborative parties, access to the data can be performed easier and cheaper, while it also opens many new opportunities such as data analytics. Electronic Health Record (EHR) system is an example of collaborations where several hospitals share medical records of patients with each other.

Next to all these benefits of sharing data across multi domain systems, there exist security concerns, particularly the risk of unauthorized access to the data. Since the shared data can be sensitive, such as medical records of patients, the data storage infrastructure should deploy proper security mechanisms to protect access to data for unauthorized users. These mechanisms need to authorize any party requesting access to the data and should provide functionalities and infrastructures to specify and enforce access policies, which are the rules specifying who can access the data.

Protecting access to the data in collaborative environments requires that all the collaborative parties use common access policies for the shared data. However, this feature increases the the complexity of a collaboration system because each collaboration domain might use different technologies and standards to control access to the data. In this case, conflicts between the technology, internal system policy that each party uses with the data policy, and ontology definitions can make the administration of such systems difficult. A trivial solution for this problem is that all the collaborating parties change the entire authorization system according to a unified technology. However this is very expensive. This highlights that addressing this drawback requires developing proper authorization mechanisms that are interoperable and are scalable. Moreover security mechanisms and part of their components and identifiers (users/resources) need to be synchronized across domains and enforced efficiently. Additionally these security mechanisms have to be able to support various granularity levels for the access to the data. More challenging is when these granularity levels (sensitivity status) change during the lifetime of the data or depend on dynamic user attributes.

Therefore the problem that we are addressing in this thesis is how to efficiently enforce access policies and how to manage conflicts of policies when data is shared across domains, while taking into account data sensitivity status.

1.2 Research Questions

The main problem addressed by this work is ***How to efficiently enforce fine-grained access policies of data shared across domains?***. Furthermore this main research question splits in two top-level research questions. Each of these top-level research questions have multiple research sub-questions that are answered in this thesis:

How to manage and enforce policies across domains efficiently?

- How to efficiently reflect collaboration access policies (defined by a central authority) in all collaboration's domains?
- How to solve conflicts between collaboration access policies and domain-specific access policies?

How to enhance the granularity level of the authorization to data based on its sensitivity status and using user attributes?

- How to handle data sensitivity when this is dynamic (e.g. based on changing legislation, policy, applications or even user demand)?
- How to enhance the access control granularity without modifying the traditional authorization solution, but by complementing it?
- How to provide fine-grained access control without affecting the performance of the application service?
- How to embed dynamic user attributes (e.g. geolocation) in access control?
  - How to handle false positives and false negatives for geolocation solutions?

1.3 Approach

In this thesis we design and implement trusted cross-domain policy enforcement components which build upon the state of the art and which address above research questions.

To answer the policy administration research question we design a system that includes architecture and required schemes to manage and enforce access policies when the data is shared across domains. We propose two layers of authorization to enforce access policies, where the first layer uses abstract collaborative policies, and the second layer enforces the domains specific policies associated with the data. The abstract policies do not include all the access rule details, but generalized policies that all the domains agree upon. This should allow easier management of conflicts between the policies from different domains, easier updating of the policies and make the evaluation of policy more efficient. This approach builds upon the solution proposed for the data sensitivity research question, which is answered by mapping resources that have to be protected to data sensitivities. The latter can be seen as abstract resources that have to be protected. By using the same abstraction technique, permissions can be mapped to abstract permissions. These abstractions lead to abstract policies.

To answer the data sensitivity question we investigate data sensitivity annotations such that services running on top of a digital platform can process, exchange, etc. the data stored by a digital platform in a compliant to its sensitivity level. For data annotations we develop an approach to annotate rows, columns and cells in a database with a sensitivity attribute or authorization policy.

For geolocation control we develop a platform service that reliably determines the country of a remote user. It allows combination of location sources , which lends itself to be policy configurable to achieve the required level of trust and tailoring to the medical use case, e.g. support of break-the-glass option.

1.3.1 Use Cases

This work is partially executed in context of the AU2EU project[1], which aims to create an e-Authentication and e-Authorization framework for collaborative delivery of services. Within the AU2EU project, the solutions designed and developed in this thesis are addressing requirements from two collaborative use cases: Translational Research (Movember) and Collaborative PACS (Philips medical image and information management system).

The aim of translational research is to aggregate patient data from different research institutes for the purpose of extracting valuable medical information about the patients. A translational research use case is Movember. Movember deals with very sensitive data, accessed from 14 areas (Research Institutes and University Medical Centres) in 5 Movember regions (Australasia, Europe, UK, Canada, USA) and by different users (biostatisticians, technicians or radiologists) in a distributed system. In this context ensuring proper access to the data is of paramount importance. Within Movember the access control granularity must be fine-grained for both resources and subjects. Furthermore the the translational research collaboration policies must be defined by the central collaboration authority. The collaboration access policies must be enforced in all translational research domains and must overrule domains specific access policies.

PACS (Picture Archiving and Communication System) is a medical imaging technology that provides storage of, and convenient access to, images from multiple modalities. Challenging from security and privacy points of view is the scenario when patient data is disclosed across different organizational and jurisdictional domains (e.g. different hospitals that host practitioners involved in teleradiology). Therefore collaborative PACS system must support access control policies for data with sensitivity levels aligned with data granularity. Collaborative PACS may be an example application service on top of a digital cloud platform. Digital cloud platforms ease providing services, online collaborations and access to data against lower costs. Architecturally, data sensitivity solutions are potential platform services to be used by other platform services and application services. For this reason reliability, practicality and user-friendliness for developers and end-users are important. The security solutions must not significantly affect user experience, must minimize impact on the system performance and must fit/complement the traditional authorization solution.

The extended description of Translational Research (Movember) and Collaborative PACS use cases can be found in appendix A, respectively appendix B. The extended security requirements list for the Translational Research use case can be found in appendix C, while the one for the Collaborative PACS use case can be found in appendix D.

[1]http://au2eu.eu/

1.4 Structure of the report

The remaining of this thesis is organized as follows:

- **Chapter 2** discusses the related work for the policy administration and enforcement, access control granularity and geolocation topics. This chapter contains also a description of collaborative traditional architecture for policy enforcement.
- **Chapter 3** addresses the policy administration research question by proposing a more efficient solution for non-conflicting policy administration. The first section analyses the gaps present in the related work. Next the proposed solution is described. Later on, the chapter contains performance analysis and experimental validation subsections.
- **Chapter 4** addresses the data sensitivity research question by proposing a data sensitivity annotations based solution for fine-grained access control. The first section introduces the objectives. Next section goes over the architectural aspects of the problem. Further on, the data sensitivity annotations solution is described, implemented and validated.
- **Chapter 5** contains conclusions and directions for future work.

Chapter 2

Related Work

This chapter surveys standards and literature related with cross-domain authorization topic. We are analysing and describing the following cross-domain authorization topics: cross-domain policy administration and enforcement, security ontologies, access control granularity (data sensitivity) and geolocation.

However, before these analyses, we describe the background information and definitions related to the state of the art. The background information includes a description of traditional architecture for collaboration authorization which can be used later as a reference on showing the impact of each related work.

2.1 Authorization Reference Architecture

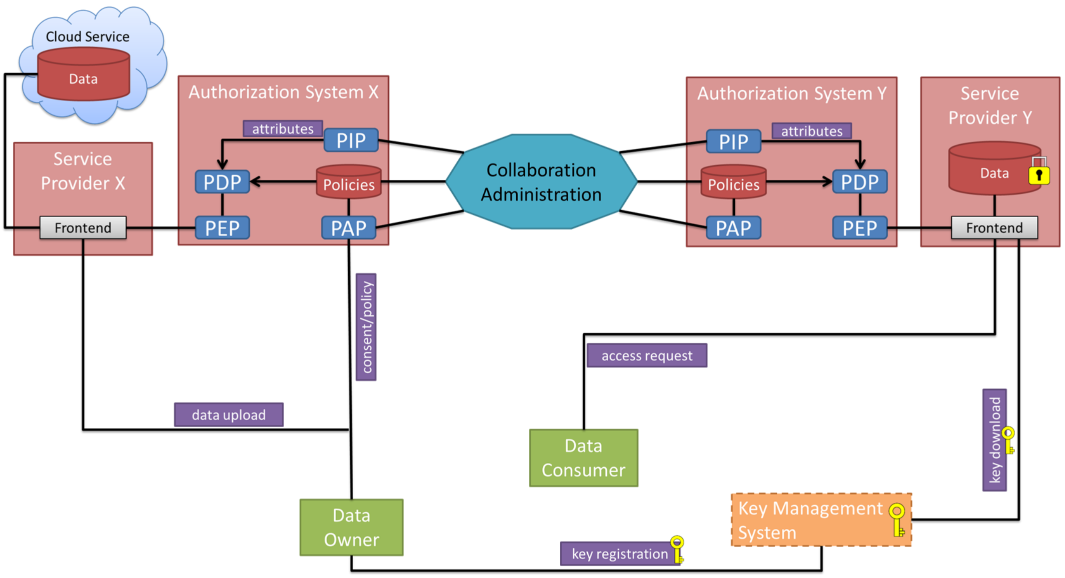


Figure 2.1: Collaborative Authorization – Traditional Architecture

The reference architecture (Figure 2.1) for authorization of actors requesting access to the data contains various components. Further on, these components are described together with their functionalities.

The **Service Provider** component is responsible for handling data management requests such as data upload or download. In addition, the Service Provider has to control the access to the data using the Authorization System component. The *Data* database might reside in the cloud or in the Service Provider.

The **Data Owner and Data Consumer** components are end users which have interfaces with the cloud service provider to upload and download data respectively. The data owner sends both, data and the policy/consent associated with the data, to the cloud service provider, who then stores the data in the database and updates the policy database via the Policy Administration Point (PAP). The data consumer requests access (e.g. read) to the data present in Service Providers database.

The **Authorization System** component has interfaces with the Service Provider and is responsible for performing access control on behalf of the Service Provider. The authorization system is a policy-based framework and consists of four core components [2]:

- **Policy Enforcement Point (PEP)**, which makes decision requests and enforces authorization decisions. PEP is used by the Service Provider whenever receives an access request to the data;
- **Policy Decision Point (PDP)**, which evaluates an authorization decision using the applicable policies retrieved from Policies database and the requester's attributes provided by PIP;
- **Policy Administration Point (PAP)**, which manages the Policies database. It is used in the upload of the users consent/policy to the Policies database;
- **Policy Information Point (PIP)**, which acts as a source of attribute values (i.e. a resource, subject, environment) and provides the PDP with required attributes;

Collaboration Management connects two Service Providers, each of them with its Authorization System. This component's main functionalities are: policies converting, ontology definition, negotiations between service providers. These functionalities have to be performed mainly at the level of and between the Policies, PAP and PIP (attributes) components.

The **Key Management System** component, which is optional, is used to store and manage decryption keys when the data owner stores data in encrypted form.

2.2 Access Control Models

This section spans over state of the art access control models. The objective of this section is to find the access control models that suit best the use cases, requirements and research questions of this thesis. The chosen access control models are used later in the design of the proposed solutions.

Discretionary Access Control (DAC) model is a way of performing access control based on identity of subjects (groups) and resources. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control). *Mandatory Access Control* (MAC) model is performing access control by restricting access to resources based on fixed security attributes assigned to both users (i.e. clearance) and resources (i.e. sensitivity). The access control is mandatory in the sense that it is enforced centrally by the system. [3]

However both DAC and MAC models rely on direct identification of the users and their association with resources. This does not optimally fit the Translational Research and Collaborative PACS use cases, because it specifies an identity rather than a sensitivity attribute, which may apply to a larger group of users (identities) or resources from which many may be unknown at the moment the policy annotation is defined.

Role-Based Access Control (RBAC) is an access control model where users are mapped to roles and roles are mapped to permissions (a permission is a tuple: $\langle action, resource \rangle$) and because of these mappings, the users get permissions which allow them to access resources [4]. RBAC can be considered an instantiation of ABAC where roles match attributes of users and resources are annotated with a role representing the required role [5]. A common representation of RBAC model is depicted in Figure 2.2.

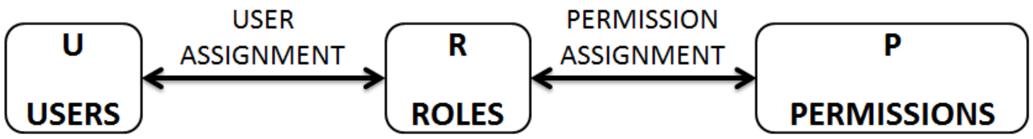


Figure 2.2: Core RBAC Model

Attribute-Based Access Control is an access control model where the access rules are defined based on users and objects attributes, without referencing specific users and objects. This allows addition of users and objects into the system without modifying the policies, with the only condition that the users and objects have attached their attributes. As such ABAC brings a benefit of simplified management [3]. Moreover ABAC facilitates the fine-grained access policy administration as per each individual subject [6].

The objectives of this thesis revolve around flexibly and efficiently dealing with data sensitivity and authorization policies management and enforcement to support a wide range of domains and applications within these domains. This fits well with the Attribute-Based Access Control (ABAC) model.

2.3 Cross-Domain Authorization

In this section we analyse the state of the art solutions related with the more broadened topic of cross-domain authorization. We are showing the impact of these state of the art solutions, their advantages and disadvantages regarding efficiency, performance and possible bottlenecks.

In order to overcome the shortcoming of traditional data security approaches, which provide solutions for the storage infrastructure only, Koster et al. [7] propose a solution for end to end security. This solution protects data from the upload to the download operations. The architecture also allows consent management, which empowers the patients to control how their health information is shared and used in a personal telehealth eco-system. The solution is based on HL7 CDA R2 Consent Directive for patient consent management. However the solution does not scale. As the number of patients grows, the number of consent documents and therefore policies increases in the system. This affects the performances of policy administration and policy enforcement. In a follow-up work Asim et al. [8] propose a solution which enforces the consent (policy) using cryptographic techniques (DEN IHE document encryption). The encryption of patient data makes the proposed solution applicable in semi-trusted domains usage model, but adds latency to the consent enforcement phase.

Another approach used in the cross-domain policy administration and enforcement is to perform authorization at the operating systems level. It makes the authorization mechanisms application and application-protocol independent. Furthermore, this approaches is able to cover better the events accessing subjects' data [1][9], compared to the traditional application-level authorization mechanisms. Kelbert and Pretschner [1] propose a policy enforcement architecture that makes use of sticky policies for distributed environments. The PIP and PMP components of this architecture communicate with their respective counterparts in other systems, while PDP and PEP components do not communicate with the exterior. However the system scalability has to suffer as the the communication between PMP and PIP components from different systems may become a bottleneck, as depicted in Figure 2.3. This bottleneck can be amplified by the number of operating system events. This is the case also for the end-to-end architecture solution (from the client to the cloud nodes) proposed by Betg-Brezetz et. al [9].

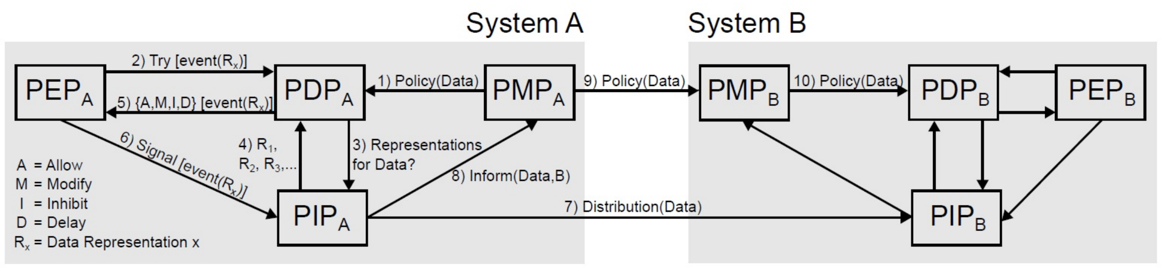


Figure 2.3: PMP and PIP components bottleneck [1]

Tang et al. [10] propose a RBAC model for multi-tenant environments. Traditional solutions to address multi-tenant access control are based on using database schema approaches. Compared to database schemas which only protect database systems, the proposed solution allows protection of both, databases and virtual machines. However, compared to traditional solutions, the model has not been extensively discussed to find its shortcomings or security holes. Even though the solution is able to protect multiple types of resources (database schemas and virtual machines), the granularity level is coarse-grained.

2.4 Cross-Domain Policy Administration

Considering the policy administration research question of this thesis, we focus in this section on solutions which are closer and closest to the across domains policy administration topic. In this topic, work has been done either from the perspective of the developer or administrator, or more focused from subject attributes point of view. Further on, adjacent to the policy administration topic is the security ontologies topic.

Han et al. [11] proposed a policy administration mechanism which assists the developer in designing policies for Android applications and end-users in verifying the policies that they are accepting upon using the applications. From an architectural point of view, this solution uses two main building blocks for definition and verification of policies. However, the text-mining algorithms used by this solution for defining policies are affecting the performance of the system.

Sahavechaphan et al. [6] proposed AAS-pada, a composition-based technique of administrating policies across multiple domains (Healthcare Providers). This technique generates specific policies on the fly by using subjects attributes and a set of generalized predefined (primitive) policies as input. The policy administration is applied in the ABAC model [5] context. However, this solution is designed for databases and is not able to protect access to other types of resources (e.g. images).

Alshehri and Raj [12] proposed The BiLayer Access Control model, with two authorization layers. The first layer is based on RBAC and uses static attributes (ABAC) for defining pseudo-roles. The second layer is based on ABAC (use of dynamic attributes only). This proposed solution addresses ABACs disadvantage of difficult policy administration. The proposed approach uses advantages from RBAC and ABAC models and diminishes their disadvantages. Even so, the evaluation criteria used in the paper show that that BiLayer Access Control model tends to preserve the disadvantages from RBAC and ABAC.

Security ontologies related solutions can also represent an answer for the policy administration problem. However security state of the art ontologies like: FOAF[1], org[2], Proton[3], OrBaC[4] revolve around the subjects, therefore are used to describe only the people, their activities, roles and relationships inside the organization/collaboration. The TAS3 Authorization Ontology[5] can be used to describe entire access control policies, covering policies, permissions, users, roles and most important resources. Even so, ontologies have often lack of intuitiveness and expressiveness [13].

Kim et al. [13] propose a solution which develops on DAML security ontology. The proposed solution facilitates mapping of higher-level (mission-level) security requirements to lower-level (resource-level) capabilities, focusing on covering all types of resources. The latter makes the solution too complex. A complex solution can insert too much latency in the system's performance. This contrasts with our requirements of designing solutions which are affecting the system's performance in a minimal way and the efficiency goal of the policy administration research question. Furthermore the proposed solution does not deal with policy conflicts.

2.5 Access Control Granularity

2.5.1 DataBase Management Systems (DBMS)

When discussing the access control granularity, the state of the art is largely formed by database management systems which offer column, row and/or cell level security access control. In these cases columns, rows or cells are subject to an access control policy, which is evaluated on access by a database user, i.e. when executing a query.

Implementation of above mentioned solutions depends on the desired granularity. Technically, it uses or adapts DBMS concepts like table structures and queries. The column level security solutions are making use of GRANT, REVOKE and DENY statements in all four.

[1]http://xmlns.com/foaf/spec/

[2]http://www.epimorphics.com/public/vocabulary/org.html

[3]http://www.ontotext.com/sites/default/files/proton-doc/proton-doc.htm

[4]http://www.ojs.academypublisher.com/index.php/jnw/article/viewPDFInterstitial/0408720733/944

[5]http://homes.esat.kuleuven.ac.be/~decockd/tas3/final.deliverables/pm36/TAS3_D07p1_IDM-Authn-Authz_V3p0.pdf

The Row Level Security (RLS) is provided in PostgreSQL using WHERE clauses for selecting the rows [14]. MySQL offers RLS by adding annotations columns in the protected tables [15]. Oracle can provide RLS using both: WHERE clauses [16] and annotations [17]. In Microsoft SQL Server, RLS is enforced using Microsoft SQL Server Views [18].

In Microsoft SQL Server, cell level security is provided with encryption and keys managed internally [18]. For PostgeSQL, Oracle and MySQL, cell level security is not supported and cannot be provided by combining row and column level security mechanisms. Column Level Security and RLS are separate mechanisms and a scenario where the user has access to the ($R1$,$C1$) and ($R2$,$C2$) cells and does not have access to the ($R1$,$C2$) cell is not possible. The supposition is that $R1$ and $R2$ are different rows in the same table and $C1$ and $C2$ are different columns in the same table.

Unfortunately, these solutions focus on enforcement - via access control and optionally encryption - by the database management system rather than enabling applications to leverage the annotations themselves for data they process and (intend to) exchange. Above mentioned solutions are user oriented (database users, not application users), modify the databases or use encryption. Thereby these solutions cannot support sensitivity annotations as intended and cannot solve the problems faced with collaborative applications.

Different from the previous solutions is Accumulo [19]. Accumulo provides cell level security by specifying "Visibility" for each key in the database tables. The disadvantage of this solution, as other RLS solutions from above is that requires big modifications of the database intended to protect. The advantage of this solution is that it started from BigTable data model [20], which is a system for managing structured data that is designed to scale to very large size. BigTable is a non-relational database (NoSQL [21]). Therefore also other non-relational database solutions can scale to very large size.

2.5.2 Data Sensitivity Annotations

Furthermore the access control granularity topic is also addressed using other mechanisms than DBMS, namely annotations. The annotations are attached to resources, subjects, policies and are used in the authorization mechanisms, sometimes next to the DBMS security solutions. Annotations can be useful because they are independent of the implementation and can be added later in the authorization mechanisms, depending on changing security requirements.

Nasirifard et. al [22] propose an annotation-based access control model for Web 2.0 social platforms. The annotations are attached to the data and they concern the subject of the policy. Using the annotations the data owner (not an administrator) decides who is able to access his data. The disadvantage is that the annotations are applicable only for the authorization subjects and not for the resources or operations.

Tran and Dang [23] propose a hierarchical video database model with annotations in order to support finer grained access control. The annotations are of various types: object, person, location, event, caption and image. The authorization mechanism, based on the content-based access control model, takes in consideration two different types of permissions: hard and soft and manages conflicts between permissions. However, the annotations are deeply embedded in the original video database and this makes the solution less adaptable for any type of database.

Li et al. [24] propose an enhancement to RBAC model based on conditions, which uses aspect-oriented programming for embedding the proposed solution in the applications, which should make it easily integrable. The proposed solution uses conditions in order to achieve fine-grained access control granularity in a multi-level approach. The administration of the authorization mechanism might not be intuitive, as conditions are used in the second level of authorization, after the classical RBAC authorization. Further on, another drawback for this solution is that conditions are textual descriptions and this might affect the integrability of the solution or induce subjectivism in the authorization framework.

Maamir et al. [25] propose a model that provides information flow access control for the object oriented paradigm based on message filtering. The solution uses access rights defined on the object attributes. This approach provides fine grained access control granularity. Further on, the solution intercepts the messages exchanged between the objects. However, message filtering is complex and can slow down the system. The solutions are deeply embedded in the application code.

2.6 Geolocation

Geolocation is a subject which is adjacent to the ABAC model. Geolocation is an attribute that is attached to the subject and that can be used in the authorization systems that uses ABAC model. In this thesis, the geolocation is used next to the data sensitivity problem, when the data sensitivity is referring to allowed geographies that data may travel to and from where application services may be accessed.

Geolocation of user (client) can be performed server-side, i.e. cloud service, based on the servers knowledge of the IP address of the client. The IP-based solutions alone provide geolocation with the accuracy of country reasonably reliably upto cities unreliably. Example geolocation services include IP2Location [26] and IPInfoDB [27]. Many more similar services are available paid or for free.

Because the accuracy of IP-based geolocation might not be enough, the geolocation information may be augmented by information from the client. The client has more information and can use various techniques in order to perform geolocation. These techniques depend on the type of device where the client resides, mobile or desktop:

- Mobile Devices: The techniques present in the mobile location based services are: GPS (Global Positioning System); Cell-phone signal tracking: Direction of Arrival (DOA), Time of Arrival (TOA) and Time Difference of Arrival (TDOA) [28]; Statistical methods [29]; Fusion of signals using neural networks [28]; Frequency drift of FM radio stations [30]. Contributions in this field are made by: Open Location Service Initiative (OpenLS), Location Interoperability (OpenLS), Internet Engineering Task Force (IETF). [31]

- Desktop: When the user is in the possession of desktop and not a mobile device equipped with GPS and other technologies, the range of technologies that can be used in detecting the location is limited. Even so this can still be done at the accuracy of country based on IP and nearby Wireless Access Points (WAPs). By sending this information to a Geolocation Service Provider, the user can be localized. Modern web browsers include support for this technology [32] [33].

2.7 Conclusions

For the policy administration research question, the solution proposed in this thesis is similar with the solution proposed by Sahavechaphan et al. [6] because both approaches use the abstract/concrete concept for the access policies. However, while the state of the art approach revolves mostly around policies instantiation and subjects attributes, our solution is about generalising resources and operations. Furthermore our solution is similar with the concept of creating an ontology for resources and operations. TAS3 Authorization Ontology [34] is able to deal with resources and permissions. While TAS3 Authorization Ontology might be able to solve the policy administration research question, the performance of the solution is at question. TAS3 authorisation ontology model is structured in classes, properties and instances. This contrasts with our solution which makes use only of mappings of objects of primitive type. The latter makes our proposed solution efficient.

For the data sensitivity research question our proposed solution makes use of annotations as done by Nasirifard et al. [22], where annotations refer to the subjects that are accessing resources. In contrast, our approach proposes data sensitivity annotations, where annotations are attached to the data (resources). Furthermore these annotations use subjects' geolocation attribute. In section 2.6 geolocation state of the art reveals techniques and mechanisms which cover the use cases requirements and challenges. Therefore these solutions are used in the proposed data sensitivity solution.

Chapter 3

Cross-Domain Policy Administration

In this chapter we answer the policy administration research question:

How to manage and enforce policies across domains efficiently?

3.1 Gaps Analysis

Nowadays collaborations share various types of *resources* (e.g. images, databases) across different domains for processing the data (image processors, analyses on database data). Because of diverse processing operations that domains perform, various types of *operations* are applied on the resources (e.g. open for images and select for databases, both are more generic read type of permissions). Next to this, collaborations also use *attributes* for identification of their users. These attributes have to be synchronized between the domains involved in the collaboration.

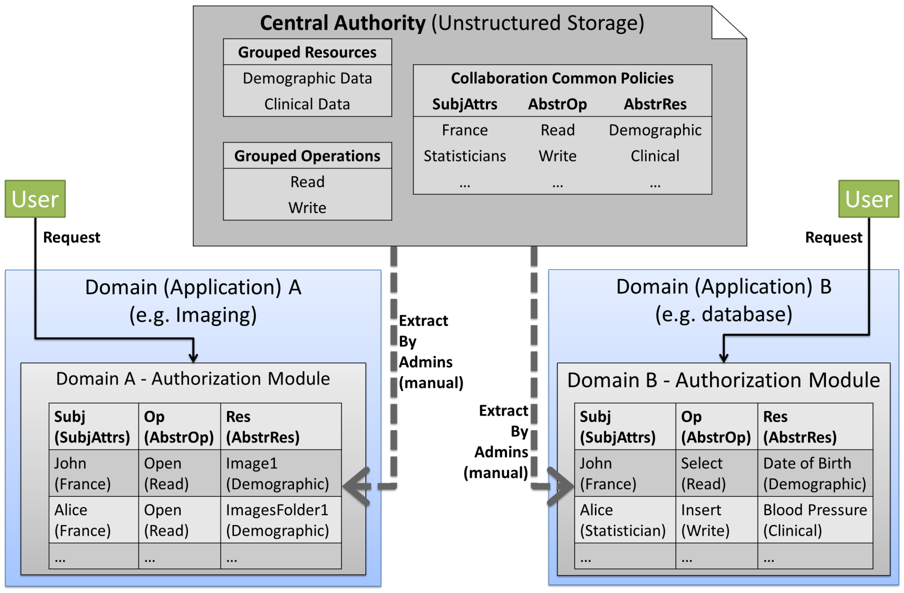
When data is shared across domains, resources are usually grouped for administration purposes using criteria like: sensitivity (e.g. High, Medium, Low), type (e.g. clinical, genomic, demographic data), institutions (e.g. Hospital A data, Hospital B data). The grouping of *resources* means mapping resources to *abstract resources* (e.g. clinical data, Hospital A data).

Also, since various types of operations by domain can be applied on data, operations are grouped into abstract operations. For example when the resources are images, the operations are *open*, *delete*, but when the resources are database elements, the operations are *select*, *insert*, *remove*. Using the *abstract operations* concept, *open* and *select* operations are grouped as *read* abstract operations; *delete*, *insert*, *remove* operations are grouped as *write* abstract operations.

In collaborations usually a central authority defines collaboration policies that storage systems and domains participating in the collaboration agree on. When these policies need to be reflected in the domains, each collaborating party extracts domain-specific policies from collaboration policies. Because the collaboration policies use abstract resources and abstract operations to specify access restrictions, we call such collaboration policies, *abstract policies*. The general format of an abstract policy makes use of abstract resources and abstract operations:

*Subjects with attributes **SubjAttributes** can perform abstract operation **AbstrOp** on abstract resource **AbstrRes**.*

In section 2.7 we concluded that state of the art policy administration and enforcement solutions cannot address in automated way with collaboration abstract policies and their abstract resources/operations. In addition, the security ontologies lack intuitiveness. Therefore, in collaborations, access policies are currently administrated and enforced as depicted in Figure 3.1. The domain-specific policies are extracted from the abstract policies by the domains administrators. The administrators of each party participating in the collaboration are using as input the documented collaboration abstract policies and replace the abstract resources/operations with the domain-specific resources/operations. For example: “A doctor located in France can read demographic data”, the administrators are identifying the images (resources) that are of demographic type and the database tables and entries that are considered demographic data. Next they are identifying the operations that mean *read* (e.g. *open* for images, and *select* for resources of database type).


Central Authority (Unstructured Storage)
User
Request
User
Request
Domain (Application) A (e.g. Imaging)
Domain A - Authorization Module
Extract By Admins (manual)
Extract By Admins (manual)
Domain (Application) B (e.g. database)
Domain B - Authorization Module

Grouped Resources
Demographic Data
Clinical Data

Grouped Operations
Read
Write

Collaboration Common Policies		
SubjAttrs	AbstrOp	AbstrRes
France	Read	Demographic
Statisticians	Write	Clinical
...	...	...

Subj (SubjAttrs)	Op (AbstrOp)	Res (AbstrRes)
John (France)	Open (Read)	Image1 (Demographic)
Alice (France)	Open (Read)	ImagesFolder1 (Demographic)
...	...	...

Subj (SubjAttrs)	Op (AbstrOp)	Res (AbstrRes)
John (France)	Select (Read)	Date of Birth (Demographic)
Alice (Statistician)	Insert (Write)	Blood Pressure (Clinical)
...	...	...

Figure 3.1: Across Domains Policy Administration - Traditional Architecture

The problems with the traditional approach of administrating policies are:

- **Low efficiency**. The big amount of access rules that are administrated and enforced in the domains authorization mechanisms are affecting the performance of the applications services that are using the authorization mechanisms.
- **Increased probability for policy conflicts** across domains:
  - Example: A user might be able to read demographic data of a patient in domain A, but he might not be able to view a demographic image related to the same patient in domain B.

  – *Policy synchronization is not a solution.* Synchronization would have to deal with different types of resources (images versus databases) and operations (*open* versus *select*).

- **Difficult to administrate** by internal stakeholders.
- **Complex to extract** by domains administrators.

3.2 Proposed Solution

In this section we propose a solution to the problem that we identified in the last section. In our approach mappings of domain-specific resources/operations to abstract resources/operations are inserted in every domain. Examples of such mappings are depicted in Figure 3.2. These mappings are created and managed by the domains administrators.

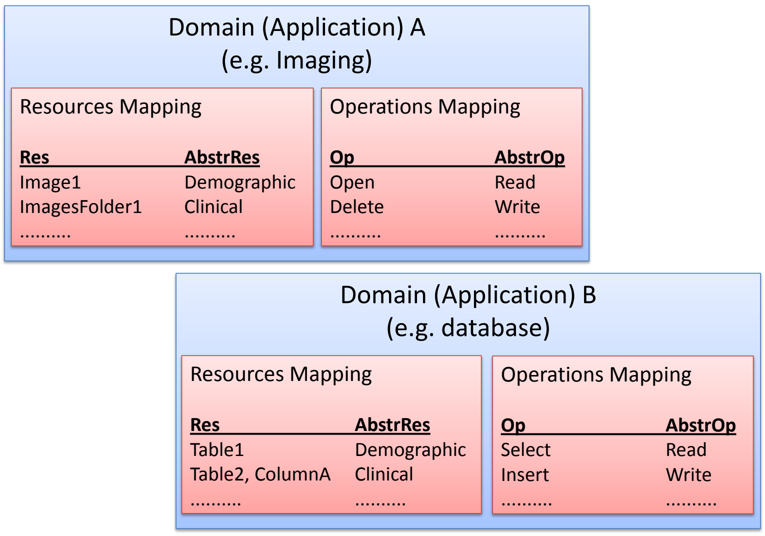


Figure 3.2: Domains Resources/Operations Mappings

Our solution is a two-layer authorization framework where the first layer contains the enforcement of the collaboration abstract policies. The high-level architecture of the proposed two-layer authorization framework, with two domains: domain A and domain B, is depicted in Figure 3.3. Each domain's PEP is abstracting the user request. Next the PEP is requesting a decision from the PDP for the newly created abstract request. The PDP is evaluating the access request from the user against the applicable collaboration abstract policies. If the decision is positive or negative, then it is enforced by the PEP component by allowing, respectively rejecting user's access to data. Otherwise, the PEP is enforcing the decision determined by domain authorization module. This module is optional and contains policies which are defined locally in the domains. Conflict resolution is facilitated by having this separation between the collaboration abstract policies and policies defined only in the domain.

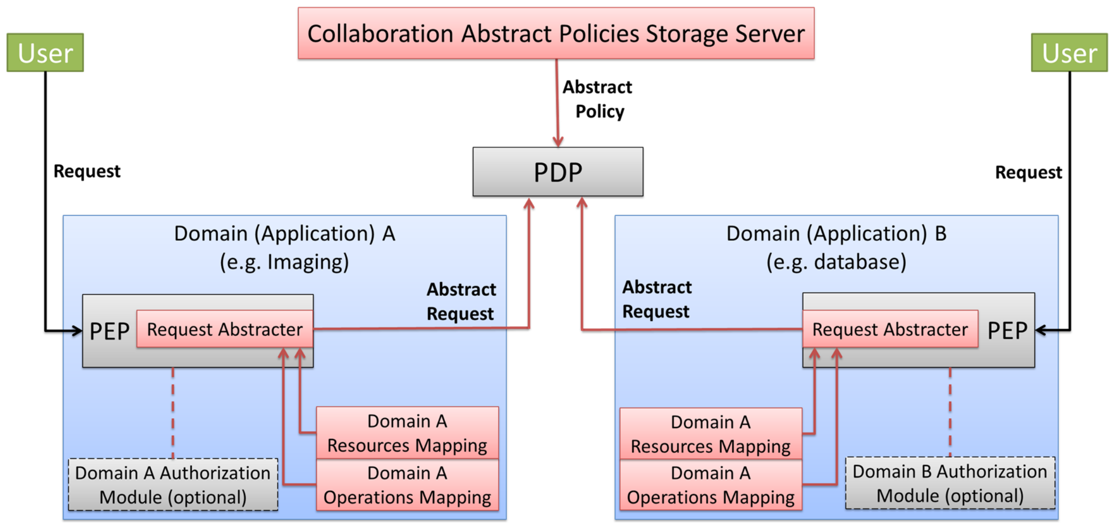

Collaboration Abstract Policies Storage Server
Abstract Policy
User
User
Request
Request
PDP
Domain (Application) A (e.g. Imaging)
Domain (Application) B (e.g. database)
Abstract Request
Abstract Request
PEP
Request Abstracter
Request Abstracter
PEP
Domain A Resources Mapping
Domain A Operations Mapping
Domain A Authorization Module (optional)
Domain A Resources Mapping
Domain A Operations Mapping
Domain B Authorization Module (optional)

Figure 3.3: Collaboration abstract authorization with two domains: Domain A and Domain B and with a central PDP

Another way of reflecting the collaboration abstract policies in the domains can be by extracting automatically the domain-specific policies. However, such a solution is similar with the traditional approach depicted in Figure 3.1 and consequently has the drawbacks identified for the traditional approach.

In Figure 3.4 are depicted the detailed steps for an user access request flow:

1. User makes a request (R) to Domain Application, which is received by PEP.
2. Request Abstracter (PEP) produces the abstract request (AR) using as input:
   (a) Received request (R).
   (b) Resources Mapping for R's resource.
   (c) Operations Mapping for R's operation.
3. PEP sends the abstract request (AR) for evaluation to PDP.
4. PDP requests applicable abstract policies from Collaboration Abstract Policies Storage.
5. Collaboration Abstract Policies Storage sends the applicable abstract policies to PDP.
6. PDP evaluates the abstract request AR against the applicable abstract policies received in step 5. Next, the PDP issues a decision and sends it to PEP.
7. Domain Authorization Module evaluates the request R against the applicable local domain policies. A decision is issued and sent to PEP. The PEP triggers this step if the decision received in step 6 is not permit or deny.
8. PEP enforces the decisions received from PDP and Domain A Authorization Module and sends to User:
   (a) Requested data, if the decision is positive.
   (b) "Access Denied" message, if the decision is negative.

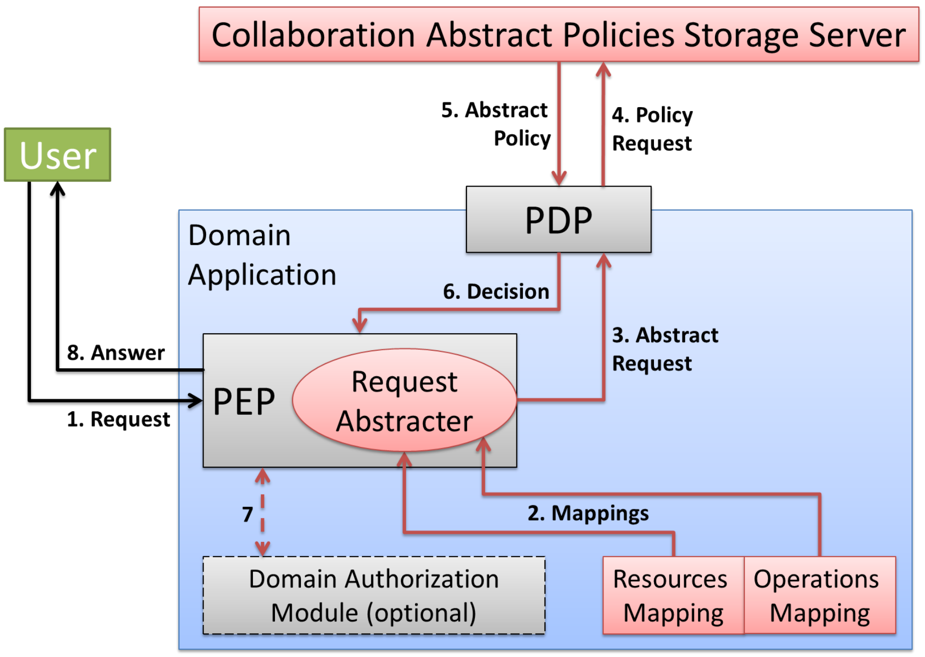


Figure 3.4: Cross-Domain Policy Administration, access request flow

3.2.1 PDP position

In Figure 3.4 it can be already seen that we placed the PDP on the border of domain. The PDP can be present in every domain or can exist only centrally. The aspects that are affected by the PDP's position are the latency generated by the communication with the Collaboration Abstract Policies Storage and the level of trust within the collaboration.

Central-PDP. When the collaboration has a central PDP significant communication latency is added to the policy enforcement process. For every request the domains' PEPs are connecting to the central PDP for every user request. Furthermore the PDP can become a bottleneck in the architecture. The reason is that the PDP has to issue decisions for all users' requests from all collaborative parties. On the other hand a central PDP may need less trust in the domains.

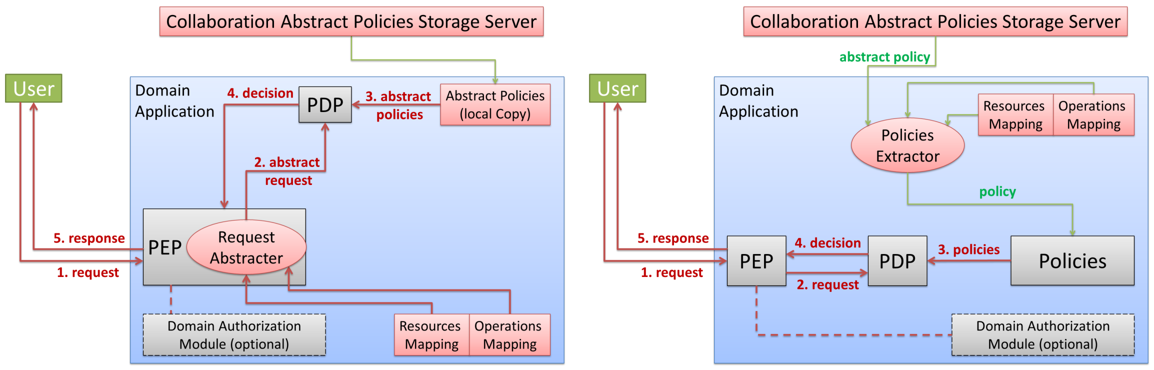
PDP-per-domain. When every domain has its own PDP the communication latency decreases, but this must be doubled by having a local copy of Collaboration Abstract Policies Storage in every domain. This situation is depicted in Figure 3.5a. The local copy of abstract policies is created in the deployment phase and updated each time the Collaboration Abstract Policies Storage is updated. On the other hand having PDPs in every collaboration domain may need more trust in the domains.

3.2.2 Performance Analysis

In this subsection we analyse the performance of Collaboration Abstract Authorization approach in comparison with the Traditional approach. Both approaches are depicted in Figure 3.5. However the traditional approach depicted in the figure is automated. This version of traditional version is needed for enabling a comparison between the two approaches.

Furthermore the comparison is even if:

- Both approaches make use of the Application Authorization Module as a separate part of the policy enforcement. This way both of the approaches are able to overrule efficiently the local domain policies. This is desirable in collaborations.
- Both approaches are making decisions (PDP) independent of any latency inserted by communications with the "Collaboration Abstract Policies Storage Server". The collaboration abstract policies are reflected in the domains by deployment (green arrows).



(a) Collaboration Abstract (b) Automated Traditional

Figure 3.5: Approaches Comparison

The notations used in the deployment performance and policies enforcement performance analyses are:

- nrR is the number of domain-specific resources
- nrO is the number of domain-specific operations
- nrP is the number of domain-specific extracted policies and this is usable only in the traditional approach, where nrP policies are extracted from the abstract policies
- $nrAR$ is the number of abstract resources from within the collaboration
- $nrAO$ is the number of abstract operations from within the collaboration
- $nrAP$ is the number of abstract policies from within the collaboration

3.2.2.1 Deployment performance

The deployment is represented in the pictures by green arrows. In order to discuss the performance of the deployment, several aspects have to be taken in consideration in both collaboration abstract and traditional approaches:

- initial policies creation
- updates triggered by a

- new domain-specific resource
- new domain-specific operation
- new abstract policy

In the next subsections we do not analyse the updates triggered by a new domain-specific operation because these updates are identical in matter of complexity analysis with the updates triggered by a new domain-specific resource.

Updates triggered by new abstract resources/operations are not considered. Firstly they imply mapping of domain-specific resources/operations to abstract resource/operations, which is alike in both approaches. Next the abstract resources/operations are used in abstract policies. This means insertion of new abstract policies, updates which are covered by “updates triggered by a new abstract policy”.

Initial policies creation

In the collaboration abstract approach the initial creation of the policies in the domain consists of downloading a local copy of the abstract policies from the Collaboration Abstract Policies Storage.

In the automated traditional approach the initial creation of the policies in the domain (*AutoTradDeploy* process) consists of downloading (*dwnld* function) each of the abstract policies from the Collaboration Abstract Policies Storage and extracting the domain-specific policies using also as input the Resources and Operations Mappings. The number of policies created per abstract policy is in average the number of domain-specific resources per abstract resource multiplied by the number of domain-specific operations per abstract operation.

$$
\begin{aligned}
C(AutoTradDeploy) &= nrAP * (C(dwnld(1)) + O(\frac{nrR}{nrAR} * \frac{nrO}{nrAO})) \\
C(AutoTradDeploy) &= C(dwnld(nrAP)) + O(\frac{nrAP * nrR * nrO}{nrAR * nrAO}) \\
C(AutoTradDeploy) &= C(dwlnd((nrAP)) + O(nrP)
\end{aligned}
$$

Updates triggered by a new domain-specific resource

In the collaboration abstract approach, when a new domain-specific resource is added in the system, this resource is mapped to abstract resources from the collaboration.

In the automated traditional approach, when a new domain-specific resource is added in the system, this resource is mapped to abstract resources from the collaboration. Next are determined the abstract policies that apply to the abstract resources previously determined. These abstract policies are translated into domain-specific policies in the domain authorization module. This is done by replacing the abstract resource with the domain-specific resource newly inserted for all the domain-specific operations that are mapped to the abstract operation from the abstract policy.

Updates triggered by a new abstract policy

In the collaboration abstract approach, when a new abstract policy is added in the collaboration, this policy is downloaded in the local copy of abstract policies from the domain.

In the automated traditional approach, when a new abstract policy is added in the collaboration, this policy is downloaded in the local copy of abstract policies from the domain. Next, domain-specific policies have to be created for all the domain-specific resources and operations (in average: $\frac{nrR}{nrAR} * \frac{nrO}{nrAO}$) that are mapped to the abstract resource and abstract operation from the abstract policy.

Deployment complexities

The complexities of the deployment phase are illustrated in Table 3.1. Figure 3.5 shows that the automated traditional approach has more components involved in the deployment processes and these are: Policies Extractor, Resources/Operations Mappings. This is similar with the work that administrators would have to do in a traditional approach. These deployment components are not present in the collaboration abstract approach.

Therefore the additional components are reflected in the complexities, which are bigger for the automated traditional approach. This makes the collaboration abstract approach more efficient considering deployment and updates of domain-specific resources/operations and abstract policies.

		Collaboration abstract approach	Automated traditional approach
Initial deployment		$C(dwnld(nrAP))$	$C(dwnld(nrAP)) + O(nrP)$
Updates	new resource (domain-specific)	$C(insertMap(1))$	$C(insertMap(1)) + O(\frac{nrAR}{nrR} * nrAP * \frac{nrO}{nrAO})$
	new abstract policy	$C(dwnld(1))$	$C(dwnld(1)) + O(\frac{nrR}{nrAR} * \frac{nrO}{nrAO})$

Table 3.1: Deployment complexities

3.2.2.2 Enforcement performance

In this subsection we compare the enforcement performance of the collaboration abstract and automated traditional approaches. In order to make this comparison, we measure the complexity of the processes that take place between the arrival of users request and the moment when the decision is issued by the PDP. Considering this, the complexities of the collaboration abstract approach $C(CollabAbstrEnf)$ and the automated traditional approach $C(AutoTradEnf)$ are:

$$\begin{aligned} C(CollabAbstrEnf) &= C(searchMap(nrR)) + C(searchMap(nrO)) \\ &+ C(decision(nrAP)) \\ C(AutoTradEnf) &= C(decision(nrP)) \end{aligned}$$

$searchMap(N)$ is the function of retrieving the abstract resources/operations for a domain-specific resource/operation. This function is applied on a mapping that contains N keys: nrR for the resources mapping and nrO for the operations mapping. This complexity depends on the way the mappings are stored and used:

- Best case scenario: $C(searchMap(N)) = O(1)$. This is happening when the mapping is loaded in the heap memory (with no hash collisions) or when the mapping is stored in a Key-Value Store (e.g. NoSQL database).
- Worst case scenario: $C(searchMap(N)) = O(logN)$. In this case the mapping is not stored or used in an optimal way, due to lack of resources. Therefore the mapping should be stored sorted. This means that the complexity becomes $O(log(N))$.

$decision(N)$ is the function of issuing a decision on N policies for a request. N is: $nrAP$ for the abstract policies and nrP for the extracted domain-specific policies. The best case scenario is when the N policies are organized in a tree where the nodes are attribute-based decision nodes. The policies' attributes are: operation type (e.g. read, write), resources attributes, subjects attributes (e.g. role, country). The complexity for the *decision* function in best case scenario becomes:

$$C(decision(N)) = O(log(N))$$

Worst Case scenario for collaboration abstract approach in comparison with traditional approach happens when *searchMap* is in worst case scenario and *decision* is in best case scenario (as $nrP >= nrAP$). Therefore, considering the above analysis, the worst case scenario for the collaboration abstract approach in term of complexities becomes:

$$C(CollabAbstrEnf) = O(log(nrR)) + O(log(nrO)) + O(log(nrAP))$$
$$C(AutoTradEnf) = O(log(nrP))$$

But:

$$nrR \leq nrP, nrO \leq nrP, nrAP \leq nrP \text{ and}$$
$$C(CollabAbstrEnf) = O(log(max(nrR, nrO, nrAP)))$$

Therefore:

$$C(CollabAbstrEnf) \leq C(traditional)$$

In real world use cases, the mapping of domain-specific resources/operations to abstract resources/operations is present in the heap memory or stored in a Key-Value Store (e.g. NoSQL database). This is happening because the mapping is small enough, respectively a NoSQL solution like Amazon DynamoDB is cheap nowadays. In this cases the *searchMap* function is optimal. This means that $C(CollabAbstrEnf)$ can become:

- If only the operations mapping can be accessed optimally:

$$C(CollabAbstrEnf) = O(log(nrR)) + O(1) + O(log(nrAP))$$
$$C(CollabAbstrEnf) = O(log(nrR)) + O(log(nrAP))$$
$$C(CollabAbstrEnf) = O(log(max(nrR, nrAP))$$

- If both the resources and operations mappings can be accessed optimally:

$$C(CollabAbstrEnf) = O(1) + O(1) + O(log(nrAP))$$
$$C(CollabAbstrEnf) = O(log(nrAP))$$

On the other hand in real world use cases the *decision* function is rarely achieving the best case scenario. This happens because of reasons like lack of heap memory or using XACML implementations with default configurations which do not fit the specific set of policies. A specific example is Sun's XACML implementation which is proven not to be optimal with the default configurations [35].

In conclusion the collaboration abstract approach is at least as efficient as the automated traditional approach in matters of *decision*. But in real world scenarios, where the *decision* is not optimal, the collaboration abstract approach is making the *decision* more efficient.

3.3 Experimental Validation

3.3.1 Implementation

To validate the efficiency of the policy enforcement of the proposed solution, this has been implemented in a proof of concept. More exactly we implemented the architectures depicted in Figure 3.5. We left out the deployment phase, which consists of retrieval of abstract policies (green arrows). The benchmarking, as the proof of concept, was developed on a server in Amazon Elastic Compute Cloud.

The proof of concept uses OASIS XACML[1] access policies. The policies are stored locally. For the policy decision point (PDP) component we used Sun's XACML[2] implementation, which is an open source implementation of the OASIS XACML standard. The resources/operations mappings are also stored locally.

For minimizing the IO operations we set the heap memory of the benchmarking application at 2560 megabytes. This way PDP loads all the XACML policies once and then it compares the received requests against them. If the heap memory is not enough, then part of XACML policies are loaded at policy decision time. This would have tampered with the tests' goal of having a fair comparison between the traditional approach and the collaboration abstract approach.

3.3.2 Validation

To validate our proposal we performed tests where we increased the amount of domain-specific resources per abstract resource. Next we performed tests where we increased the amount of abstract policies. The tests are characterized by the parameters illustrated in Table 3.2.

In the test scenarios the subjects (users) have two attributes: country and role. The country attribute has 28 possible values (the European Union countries) and the role attribute has 10 possible values.

Furthermore we generated and stored XACML policies as a set. Therefore the policies are stored in a tree, where the first decision node contains the abstract resource. This way the policy decision becomes more efficient and is closer to the most efficient solution which happens when the policy sets and policies are optimally clustered and reordered [35].

In these policies, by default, the access is denied and policies which permit access are defined. In this scenario the maximum number of policies is half of $nrSubjAttrsValues *$ $nrAO * nrAR$. If the half is exceeded then it is more efficient to switch and use by default

[1]`https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`

[2]`http://sunxacml.sourceforge.net/`

Parameter	**Value**	
	Increasing amount of domains-specific resources per abstract resources	**Increasing amount of abstract policies**
Abstract operations	2	
Domain-specific operations	2	
Abstract resources	10	
Domain-specific resources	3 - 40	20
Maximum amount of abstract policies ($maxAmountAbstrPolicies$)	$5600 = 10 * 2 * 10 * 28$ (10 abstract resources, 2 abstract operations, 10 roles, 28 countries)	
Abstract policies	25% of $maxAmountAbstrPolicies$	1%-50% (step: 1%) of $maxAmountAbstrPolicies$

Table 3.2: Policy administration, tests parameters and values

permit and to add policies which deny access. This way the total number of policies is minimized to 50% of the maximum number of policies.

3.3.2.1 Abstract policies enforcement - amount of domain-specific resources

In this test we increased the amount of domain-specific resources per abstract resource and we measured the duration (in milliseconds) of computing the decision for a user request. Figure 3.6 depicts the mean plus/minus the standard deviation of the measurements. The plot shows that enforcing abstract policies by the collaboration abstract authorization is more efficient than the traditional authorization. The non-linearity of traditional approach curve is caused by the increased amount of extracted domain-specific policies. This is caused by the increase of the amount of domain-specific resources.

3.3.2.2 Abstract policies enforcement - amount of abstract policies

In this test we increased the amount of abstract polices and we measured the duration (in milliseconds) of computing the decision for a user request. Figure 3.7 depicts the mean plus/minus the standard deviation of the measurements. The plot shows, again, that enforcing abstract policies by the collaboration abstract authorization is more efficient than the traditional authorization.

This time both curves are linear. Even though the number of policies is increasing, the number of abstract resources and domain-specific resources is not. As established above, the resources are the first criterion in searching of policies relevant for the user request. This makes the policy searching and therefore the policy decision less sensitive to the increase of amount of policies.

The variability of the durations, in both tests, shows that the traditional approach is less stable in comparison with the collaboration abstract authorization. This is mainly caused by the Sun's XACML implementation, which proves to be rather unstable when dealing with a

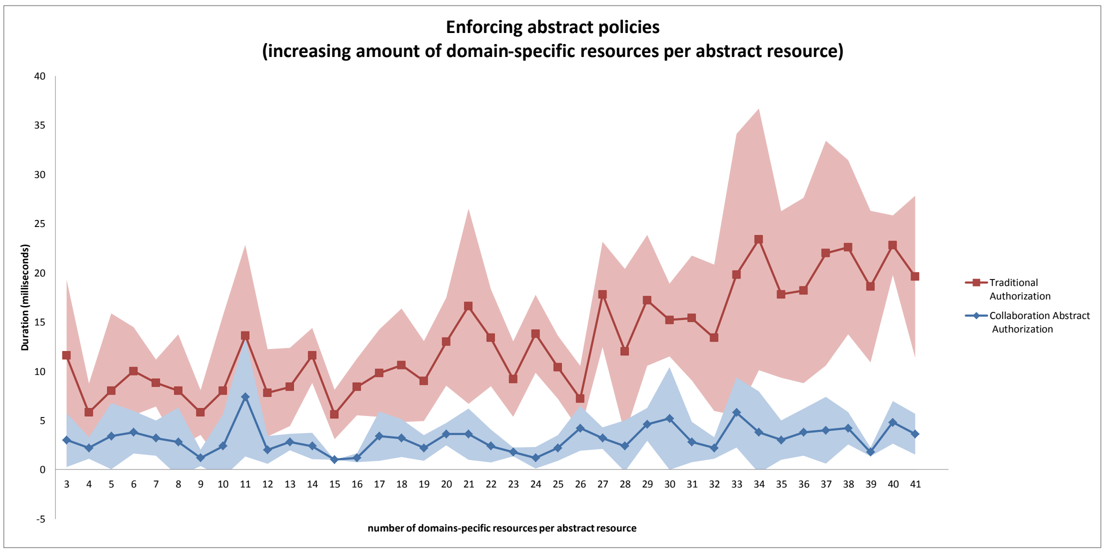


Figure 3.6: Enforcing abstract policies (increasing amount of domain-specific resources per abstract resource)

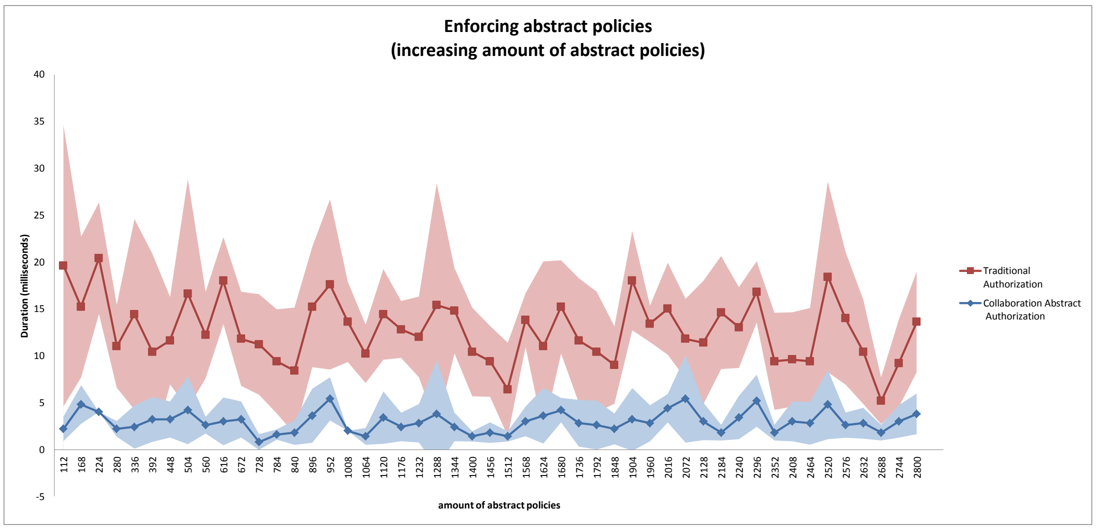


Figure 3.7: Enforcing abstract policies (increasing amount of abstract policies)

big amount of policies.

Chapter 4

Data Sensitivity

In this chapter we answer the data sensitivity research question:

> ***How to enhance the granularity level of the authorization to data based on its sensitivity status and using user attributes?***

4.1 Objectives

The related work presents solutions with security mechanisms embedded in the database. Such solutions contrast with the main requirement that it should be easy to adopt the security solutions in the application services. Preferably no constraints or modifications apply to the database used by the application service, e.g. the database schema. Data sensitivity should be signaled at application level. Therefore, we propose a solution based on annotations, which sit next to the application database tables.

Data sensitivity annotations address the need to manage (fine-grained) data over its life-cycle in compliance with its sensitivity status and applicable policies. One business use case is to manage data export control on a per-country or per-region basis. This addresses the gap of data(base) management systems, which just support discrete control access. Rather, attributes of data and attributes of the user or the intended use matter. Important requirements are flexibility, performance and manageability.

Geolocation control addresses the need to control enforcement of allowed geographies data may travel to (e.g. privacy legislation) and from where application services may be accessed from (e.g. regulatory clearance). Requirements for geolocation control include a good balance between usability, security, performance and infrastructure support (e.g. mobiles with apps and PCs with browsers).

Data sensitivity annotations are intended to deal with a number of situations:

- Certain data not to leave certain countries
- Categorical reclassification of data through legislation changes with implied change in protective measures (e.g. from clear text to segmentation, encryption)
- Data obtained under certain restrictions and conditions (e.g. privacy / consent policy)

Here we target data at its largest level of granularity, i.e. field level or cell level in database terms.

The consent policy mentioned under the third bullet stretches naturally into user-defined authorization policies for data usage and sharing, which consequently may stretch the solution from pure sensitivity into policy annotations.

Furthermore data sensitivity annotations should exploit the Attribute-Based Access Control paradigm. This translates into including users attributes (e.g. role, geolocation) in the annotations.

4.2 Architecture

Figure 4.1 represents an architecture for data sensitivity. Key component is the authorization platform service with annotation engine for data sensitivity functionality, which supports application services (e.g. PACS) adhering to the data sensitivity annotations. Both application and authorization services interact with database platform services, e.g. relational (SQL) and other types of databases.

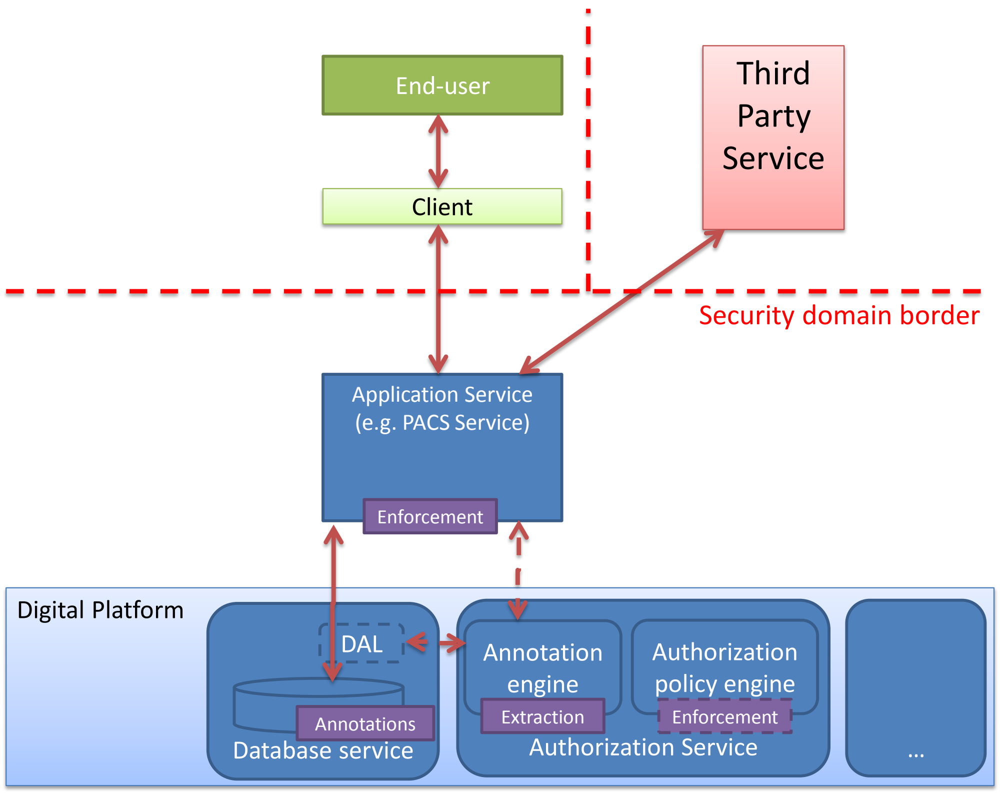


Figure 4.1: Data sensitivity architecture

A question of significant architectural impact is how to obtain sensitivity annotations for queried data. A transparent solution has advantages for application developers. One way is a kind of Data Abstraction Layer (DAL) in front of the database service, which makes it possible to determine if sensitive (annotated) data was involved in the query. Alternatively, it is based on mechanisms to log executed queries.

The annotation engine extracts relevant annotations from information received from the DAL, e.g. the query, query results and/or log data. This is a non-trivial process. A basic approach could support column annotations based on the select clause, etc., in non-complex queries. More complex cases require more advanced algorithms. It is assumed such DAL-like functionality will be provided to application service developers in form of an SDK next to access of the authorization platform service API. Figure 4.3 presents a more detailed architectural view of this.

Decisions and enforcement of decisions based on sensitivity annotations (policies) for above mentioned purposes are the primarily responsibility of the application service with more classic and generic support provided by the authorization service (platform). From a security perspective this division of responsibilities for enforcement may be distributed this way, because both components belong to the same administrative security domain as depicted in the figure.

The architecture allows annotations to be used in three ways:

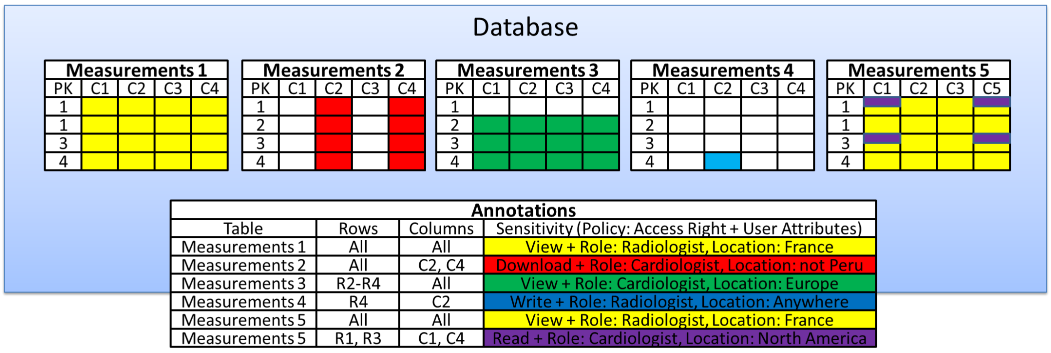
- Enforcement by application services, e.g. determine if data may be shared with a client or other third party service.
- Logging for compliance purposes, e.g. to audit compliance or in case of changing legislation determine application services, database tables or sensitivity annotations requiring updating.
- Classic access control by authorization service policy engine, e.g. if an end-user is entitled to the data or the application service purpose matches the sensitivity annotation.

The data sensitivity solution is complementary to the authorization functionality that is already in place. For example the classical authorization system controls the access to the data at the level of database, access to a service, etc. Data sensitivity annotations can also be used by the authorization system to refine and protect better the access to the data, e.g. sharing with third parties.

4.3 Data sensitivity annotations

4.3.1 Annotations management

Figure 4.2 shows how a digital cloud platform and its application services can annotate its data with sensitivity policies.



Measurements 1

PK	C1	C2	C3	C4
1	🟨	🟨	🟨	🟨
1	🟨	🟨	🟨	🟨
3	🟨	🟨	🟨	🟨
4	🟨	🟨	🟨	🟨

Measurements 2

PK	C1	C2	C3	C4
1		🟥		🟥
2		🟥		🟥
3		🟥		🟥
4		🟥		🟥

Measurements 3

PK	C1	C2	C3	C4
1				
2	🟩	🟩	🟩	🟩
3	🟩	🟩	🟩	🟩
4	🟩	🟩	🟩	🟩

Measurements 4

PK	C1	C2	C3	C4
1				
2				
3				
4		🟦		

Measurements 5

PK	C1	C2	C3	C5
1	🟪🟨	🟨	🟨	🟪🟨
1	🟨	🟨	🟨	🟨
3	🟪🟨	🟨	🟨	🟪🟨
4	🟨	🟨	🟨	🟨

Annotations

Table	Rows	Columns	Sensitivity (Policy: Access Right + User Attributes)
Measurements 1	All	All	View + Role: Radiologist, Location: France
Measurements 2	All	C2, C4	Download + Role: Cardiologist, Location: not Peru
Measurements 3	R2-R4	All	View + Role: Cardiologist, Location: Europe
Measurements 4	R4	C2	Write + Role: Radiologist, Location: Anywhere
Measurements 5	All	All	View + Role: Radiologist, Location: France
Measurements 5	R1, R3	C1, C4	Read + Role: Cardiologist, Location: North America

Figure 4.2: Database with data sensitivity annotations

Annotations may be stored in a relational (SQL) database or a BigTable system may be leveraged to optimize performance of the annotations table for lookup and searching in case it grows large. An annotation contains two parts: identification of the protected data and policy annotation (contained in the sensitivity column).

4.3.1.1 Identification of annotated data

Data identification is based on table, column and row identifiers. The table identifier must be present in all annotations. Data identification without Table identifier is not possible. Columns are identified based on their names. Rows are identified based on the primary key of the belonging table.

Alternatively, the identification of the rows could be done based on the data's attributes. This would imply leveraging database WHERE clauses in the process of identifying the protected data. This conforms to the attribute-based access control paradigm which makes use of data attributes. Moreover this solution would allow the newly inserted database entries (rows) to have automatically attached policies.

Data sensitivity annotations at cell level is future work. The first step is identification of data at cell level. This should be done using the tuple $\langle row, column \rangle$ for identifying a cell. The row and the column from the tuple are identifiers explained above.

4.3.1.2 Policy annotation

Policy annotations are contained in the sensitivity column. Data sensitivity is translated to a policy annotation. This contains the users attributes (e.g. role, geolocation) and the action that the user is allowed to apply on the data.

Leveraging the role-based access control paradigm is done by leveraging attribute-based access control. Figure 4.2 contains in the sensitivity column the attribute role, which contains values like: cardiologist and radiologist.

The action that the user is allowed to apply on the data is also contained in the policy annotation. Besides the examples in Figure 4.2: read, write, download, the action can also be export or store. Such an action would enable management of data export control (or storage) on a per-country or per-region basis. For this data annotations should be combined with the geolocation service in order to control the destination of the data.

4.3.2 Annotations enforcement

From architectural point of view data sensitivity annotations are enforced using a Filter component which sits in the Data Abstraction Layer (DAL). This can happen in the application service or in the platform service. In Figure 4.3 the Filter is in the application service.

Requests from the presentation layer are processed as usual in the application (black arrows). However, instead of directly returning the requested data to the presentation layer, DAL masks the requested data using the Filter component. This component retrieves from the annotations storage the annotations applicable to the requested data. Furthermore the Filter enforces these annotations and masks the requested data. The masked requested data is returned to the presentation layer.

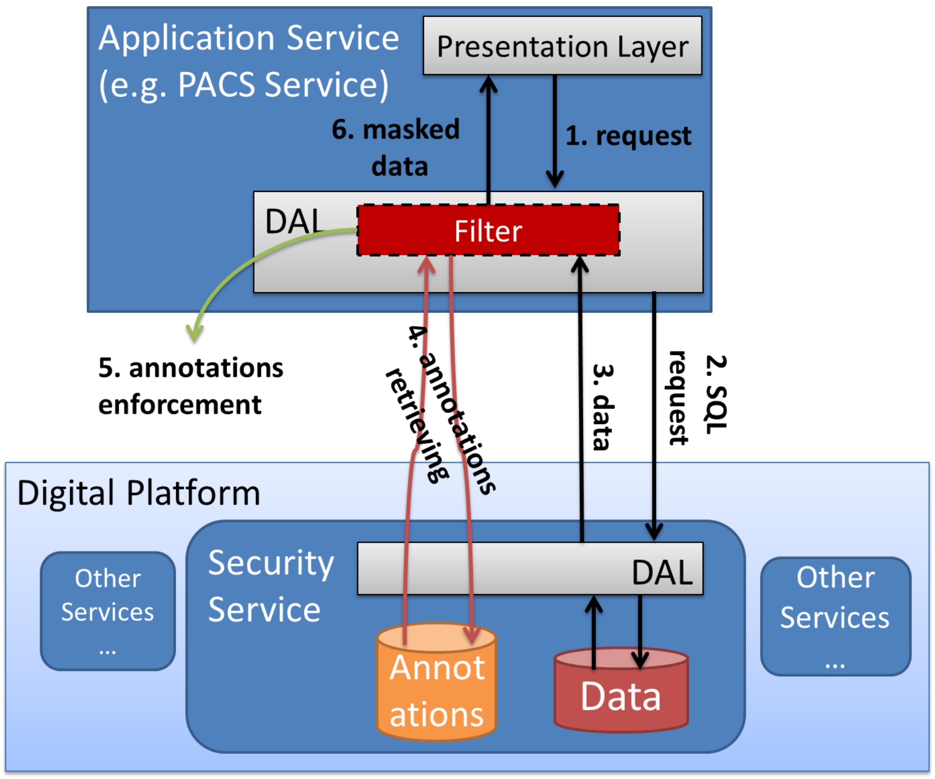


Figure 4.3: Data Sensitivity Annotations Enforcement

When the original request (black arrows) does not contain the primary key from tables of the requested data, then the request is modified with the needed primary keys and re-executed. This must be done in order to identify the rows.

As expected, additional queries affect performance. However, this does not have to make the retrieval of masked requested data twice slower, because the database cache will facilitate - confirmed experimentally - a faster execution of the modified SQL request. Another aspect is that the data transfer from the service (e.g. PACS service) to the user also introduces data transfer latency and makes the annotations retrieval and enforcement to weigh less in whole process of using the service (e.g. PACS service).

Considering the design of the enforcement, two of the approaches can be used in implementation for policy representation and evaluation:

- Proprietary policy representation implementation. Data sensitivity annotations are stored in a table/storage. The annotations belonging to the requested data that has to be masked are retrieved from the database. The results (action and users attributes) are further processed in order to compute the mask for the requested data.

- XACML implementation. Data sensitivity annotations are stored in form of XACML policies. The data identifiers (Table, Columns, Rows) are resources attributes in XACML policies scenario. This fits with XACML standard (Attribute-Based Access Control). Therefore an implementation of XACML might be used in data sensitivity annotations solution.

In both cases the relevant policies are obtained via the method depicted in Figure 4.3

4.3.3 Data masking solution design

In this subsection we propose a solution which complies with the solution directions specified above. Also other solutions may be designed, which may fit better other scenarios.

4.3.3.1 Data masking process

The process designed to mask the data comprises of five actions that are executed:

1. Execution of original SQL request (ExecSQLReq)
2. Execution of modified SQL request (ExecModfdSQLReq). This SQL request contains in addition to the original SQL request the primary keys of the tables from where the data comes from.
3. Getting annotations (GetAnnttns). This component is retrieving the annotations related to the requested data: the operation and the requester (subject) attributes from a Key-Value Store (NoSQL database).
4. Computing the mask (CompMask). This component computes the list of rows and columns that the user has access to. The resulted list represents the mask.
5. Applying the mask (ApplyMask). This component applies the mask computed before on the retrieved data in ExecSQLReq process component.

By default access is denied and policies (operations & subjectAttributes mapped to resources) which permit access have to be defined. In this scenario we assume that the maximum number of policies is half of $nrSubjAttrsValues * nrOp * nrRes$. If the half is exceeded then it is more efficient to switch and use by default permit and to add policies which deny access. This way the total number of policies is minimized to 50% of the maximum number of mappings (*operation* & *subjectAttributes* mapped to a resource).

The annotations are associated to a resource ($dataIdentifier$), in our case columns and rows. This would suggest that we should store the policies in a map where the keys are the $dataIdentifiers$ (table & columns/rows) to annotations (*operation* & *subjectAttributes*). We prove later that this would not be efficient for retrieving annotations from the database. Therefore we stored the annotations by mapping *operation* & *subjectAttributes* to a list of $dataIdentifiers$.

4.3.3.2 Performance and complexity analysis

In the performance analysis we will make use of the next notations:

- nrR: total number of database rows
- nrC: total number of database columns
- fC: selected fraction of the database columns
- fR: selected fraction of the database rows

- $sizeWhereClauses$: size of the criteria from the original SQL request
- $nrAnnttnsOpSubj$: number of annotations associated with the current user
- Pk: Primary Key
- $DataTransfFromDP$: data transfer from Digital Platform to Application Service
- $DataTransfFromAppS$: data transfer from Application Service

The complexity of the retrieving of masked data process is given by the sum of the complexities of the process components:

$$\begin{aligned}C(maskData) = {} & C(ExecSQLReq) + C(ExecModfdSQLReq) + C(GetAnnttns) \\ & + C(CompMask) + C(ApplyMask)\end{aligned}$$

Next we analyze the complexities of each of the retrieving of masked data components:

1. **ExecSQLReq**. This process component has two other internal components: the searching of the data in the database and the transfer of the data from the digital platform to the application service:

$$\begin{aligned}C(ExecSQLReq) &= C(SearchData) + C(DataTransfFromDP) \\ C(ExecSQLReq) &= O((fR * nrR) * (fC * nrC) * sizeWhereClauses) \\ &\quad + C(DataTransfFromDP) \\ C(ExecSQLReq) &= O(nrR * nrC * sizeWhereClauses) + C(DataTransfFromDP)\end{aligned}$$

2. **ExecModfdSQLReq**. This process component is the same as the $ExecSQLReq$ component, but the searching of the data takes advantage of the database cache. Therefore the complexity of searching the data in the database becomes constant and in consequence the complexity of $ExecModfdSQLReq$ component becomes:

$$\begin{aligned}C(ExecModfdSQLReq) &= C(SearchData) + C(DataTransfFromDP) \\ C(ExecModfdSQLReq) &= O(1) + C(DataTransfFromDP)\end{aligned}$$

3. **GetAnnttns**. The complexity takes advantage of the fact that we chose to store the annotations in a Key-Value Store and the complexity becomes constant.

$$C(GetAnnttns) = O(1)$$

4. **CompMask**. This component creates on the selected data two mappings: columns names to columns indexes and rows primary keys to rows indexes. Next the allowed resources are added in the allowed list of resources (mask), by making use of the mappings previously used. Mappings are used because when properly implemented they can achieve complexity $O(1)$ for the operation of searching.

$$\begin{aligned}C(CompMask) &= O(fC * nrC) + O(fR * nrR) + O(nrAnnttnsOpSubj) \\ C(CompMask) &= O(max(fC * nrC, fR * nrR, nrAnnttnsOpSubj) \\ C(CompMask) &= O(max(nrC, nrR, nrAnnttnsOpSubj))\end{aligned}$$

5. **ApplyMask**. The main operation in this component is going through the retrieved data by the *ExecSQLReq* component and applying the mask computed by the fourth process component.

$$C(ApplyMask) = O(fC * nrC * fR * nrR)$$
$$C(ApplyMask) = O(nrC * nrR)$$

As specified before we are also analysing the complexity of getting annotations from a Key-Value Store where the key is the data identifier and the value is a list of *operation* & *subjectAttributes*. In this scenario the annotations are requested (from Key-Value Store) for each requested resource (column or row).

$$C(GetAnnttnsDataIdKey) = (fC * nrC + fR * nrR) * O(1)$$
$$C(GetAnnttnsDataIdKey) = O(fC * nrC + fR * nrR)$$
$$C(GetAnnttnsDataIdKey) = O(nrC + nrR)$$

Therefore the operation of retrieving annotations from a Key-Value Store where the key is the data identifier is more expensive than the version where the key is *operation* & *subjectAttributes*. The other process components have the same complexity in both annotations storage scenarios.

Retrieving Masked Data Process Component	**Complexity**
Execution of original SQL request	$O(nrR * nrC * sizeWhereClauses) + C(DataTransfFromDP)$
Execution of modified SQL request	$O(1) + C(DataTransfFromDP)$
Getting annotations	$O(1)$
Computing Mask	$O(max(nrC, nrR, nrAnnttnsOpSubj))$
Applying Mask	$O(nrC * nrR)$

Table 4.1: Complexity Analysis

Table 4.1 shows that the most demanding process component is the execution of the original SQL request. If there are no “WHERE” clauses then the execution of the original SQL request is equal to application of the mask operation. Even so the execution of original SQL request contains also the data transfer from the digital platform latency.

The complexities analysis cannot show precisely how expensive the data transfer is. Strictly speaking data transfer is not part of retrieving and enforcing the annotations, but it is a significant part of the total latency for user experience, which makes retrieval of masked data close in matter of time complexity to the retrieval of unmasked data. Data retrieval time is roughly linear to data size.

4.3.4 Geolocation

4.3.4.1 Objectives

Geolocation control is intended to deal with the following issues:

- Data that may not leave a certain country or countries, e.g. for privacy legislation;
- Device enablement on a per-country basis. Application services may not be offered to certain countries or markets for managing regulatory clearance or because of marketing reasons.

Geolocation control for data may integrate to authorization and also sensitivity annotations discussed above through an geolocation attribute of the user/subject (e.g. in the Attribute-Based Access Control paradigm)

The geolocation control security model assumes semi-trusted users, e.g. employees or parties such as customers with which a contractual relationship exists. The geolocation control therefore mostly is intended to keep honest people honest and to ease compliance rather than attempting to achieve an impossible 100% security. This is an intrinsic technical limitation of location control.

Geolocation by definition has a margin of error, which may result in false positives and false negatives. False negatives are inconvenient for the client because he may not access the service or obtain the requested data. False positives may inappropriately make the service or data available to unauthorized geographic locations and thereby e.g. violate privacy legislation. A balance is required.

Geolocation should not constitute a hard barrier for the user access to the service or its resources. If geolocation is unavailable the default will be to allow user access. Therefore, when the location of the user cannot be decided because of services being down or contradicting, the geolocation should not be taken as input in the authorization decision.

4.3.4.2 Architecture and process

Figure 4.4 presents an architecture for a digital cloud platform with geolocation platform service. The geolocation service exposes an API towards application services that require geolocation control. The geolocation service may also interact with the authorization service for enforcement. Both for efficiency and cost reasons the geolocation service employs a cache.

Optionally, also clients such as mobile applications, desktop applications and web browsers include geolocation functionality, which assist the geolocation service in determining the location of the user of the client. In certain situations end-users may interact with the geolocation service via their client to deal with exceptions.

The geolocation platform service is supported by third party. The same holds for client geolocation support. These services translate IP-addresses, GPS coordinates, radio beacon proximity, etc., to normalized locations, e.g. addresses, including country.

To determine location of the user the components execute the following process. The application service or a common platform service like authentication extracts the IP address belonging to the request of the client. The service invokes the geolocation application service with the IP address as parameter. The geolocation service checks its cache for a match. In case of no hit it invokes a third party IP geolocation service.

Above steps yield a location (may be ‘unknown’), accuracy (‘country 98% accurate’) and trust (‘trusted service’) indication. If sufficient the process is completed, otherwise it may continue to improve.

Geolocation service may continue to send a request to the client for its location. The related work section described how mobile apps and desktop browsers may determine location. This results in a location and accuracy indication returned to the geolocation service, which

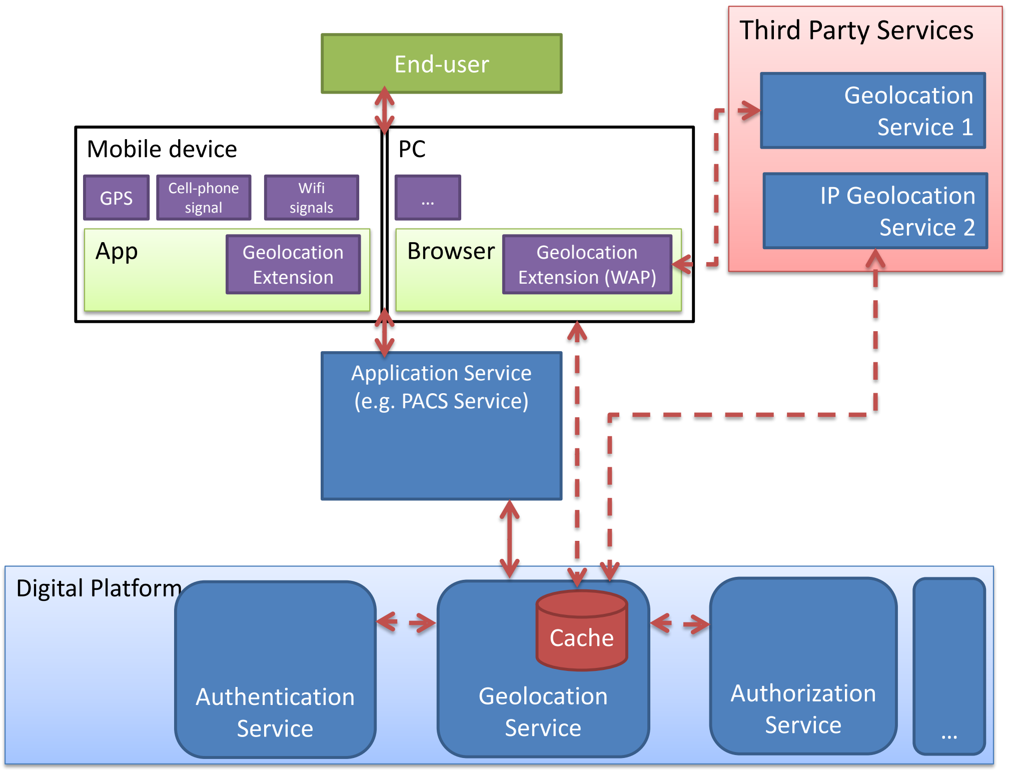


Figure 4.4: Geolocation architecture

adds the (lower) trust indication depending on the client. If sufficient the process is completed, otherwise it may continue.

Finally, the geolocation service may continue to involve the authenticated end-user to determine the location if the client supports it, e.g. by presenting a web page in the users browser. It requests the user to provide his location or to confirm / deny a certain location. It also informs the user the provided information may be subject to audit. This results in a location indication returned to the location service which adds a (lower) trust indication. Optionally it queues it for audit. Geolocation determines if it accepts or denies the provided location in which it may use all information obtained as part of above steps. Optionally, in case of denial it offers the user with a break-the-glass functionality (with default audit).

Geolocation service returns location and accuracy to the requesting application (or platform) service. It also caches the results.

The supported location sources and threshold configuration may be policy configurable.

4.4 Experimental Validation

4.4.1 Implementation

To validate the architecture and design these have been implemented in a proof of concept. This includes an application service implemented using Java Servlet and JavaScript technologies. This web application resides in our case on a server in Amazon Elastic Compute Cloud. The setup was such as to represent a realistic deployment.

The data is stored in and retrieved from Amazon Relational Database Service (RDS, MySQL). The annotations are stored in a different server in Amazon Elastic Compute Cloud, but on a different server than our application service. Initially, the same database was used, but after various performance optimizations a nosql solution in form of DynamoDB has been selected for the annotations.

For performance optimization also the table structure and queries have been rearranged for performance, i.e. reducing the number of queries, enable use of indices and avoid expensive operations like text matching. It is expected that further optimizations are possible as the optimization process was stopped when performance was sufficient to determine positive feasibility.

4.4.2 Validation

To validate our proposal we performed tests where we increased the amount of data that is retrieved from the database. Next we performed tests where we increased the amount of annotations that are attached to the data. The tests are characterized by the parameters illustrated in Table 4.2

Parameter	**Value**	
	Varying amount of database entries	**Varying amount of annotations**
Total amount of database columns (nrC)	40	
Total amount of database entries (nrR)	200 - 20000 (step: 200)	12000
Amount of retrieved columns	1/2 of total amount of database columns	
Amount of retrieved rows	1/3 of total amount of database entries	
Maximum amount of annotations ($maxAmountAnnttns$)	$(nrR + nrC) * 2 * 10 * 28$ (2 operations: read and export, 10 roles and 28 countries)	
Amount of annotations	10% of $maxAmountAnnttns$	1%-25% (step: 1%) of $maxAmountAnnttns$

Table 4.2: Data sensitivity annotations, tests parameters and values

In the test scenarios the subjects (users) have two attributes: *country* and *role*. The *country* attribute has 28 possible values (the European Union countries) and the *role* attribute has 10 possible values. The resources are represented by the combinations: $table\_columnName$ or $table\_rowPkValue$. Therefore the number of resources is $nrC + nrR$.

4.4.2.1 Retrieving data test - data size

In this test the amount of retrieved data was varied and we measured the duration (in milliseconds) of retrieving up to 7000 database entries: masked and unmasked. It can be seen in Figure 4.5 that the retrieval of masked data is more expensive than the retrieval of unmasked data, but that the performance of retrieving masked data is close to the performance of retrieving unmasked data.

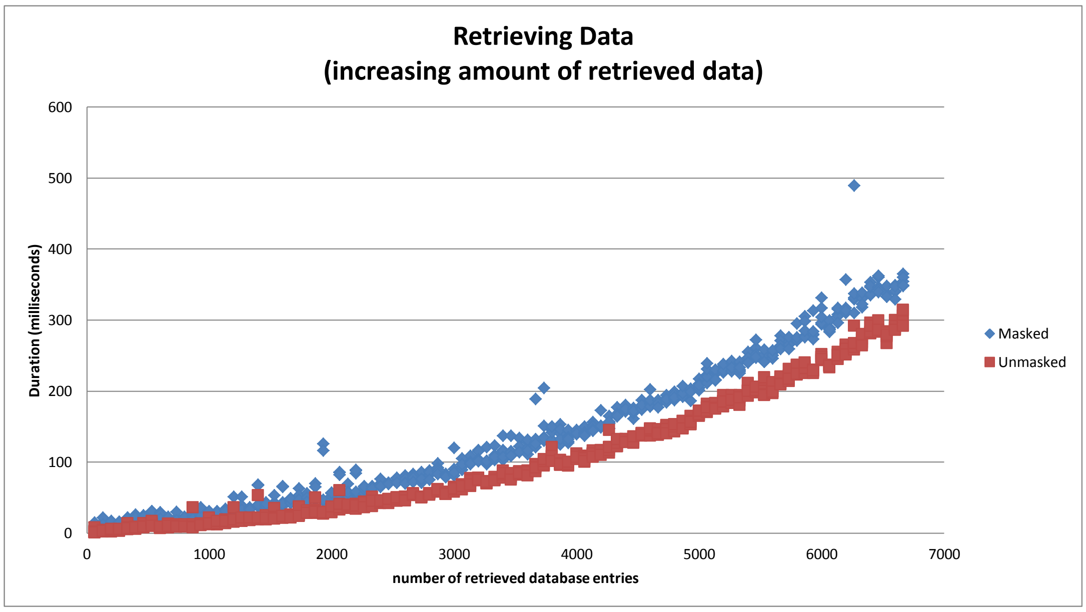


Figure 4.5: Retrieving Data (increasing amount of retrieved data)

When the data transfer (from application service) latency is added (see Figure 4.6), it can be seen that the red curve (retrieval of unmasked data) is almost covering the blue curve (retrieval of masked data), making the curves even closer than the above test. In practice, the data transfer (from application service) latency is part of the user experience, when the application service is used directly by other applications or by the users. Therefore the retrieving and enforcement of annotations affects the retrieval of data in a minimal way.

The non-linearity of the curve is caused by the SQL request where clause has the size equal with the amount of retrieved rows and looks like this: "where id in (value1, value2, ..., valueN)".

4.4.2.2 Masking components weights test - data size

In this test the amount of retrieved data was varied and we measured the weights of the components of the retrieving masked data process. The plot depicted in Figure4.7 shows that with the increase of the amount of retrieved data (up to 7000 database entries), the summed up weight of the specific masking components (ExecModfdSQLReq, GetAnnttns, CompMask and ApplyMask) decreases, achieving less than 20% out of the whole process. This means that the retrieving and enforcement of annotations counts less in the whole process of retrieving masked data with the increment of the amount of retrieved data.

4.4.2.3 Retrieving data test - annotations size

In this test the amount of annotations attached to the data was varied and we measured the duration (in milliseconds) of retrieving 4000 database entries: masked and unmasked, with up to 25% of the maximum number of annotations attached. It again can be seen in

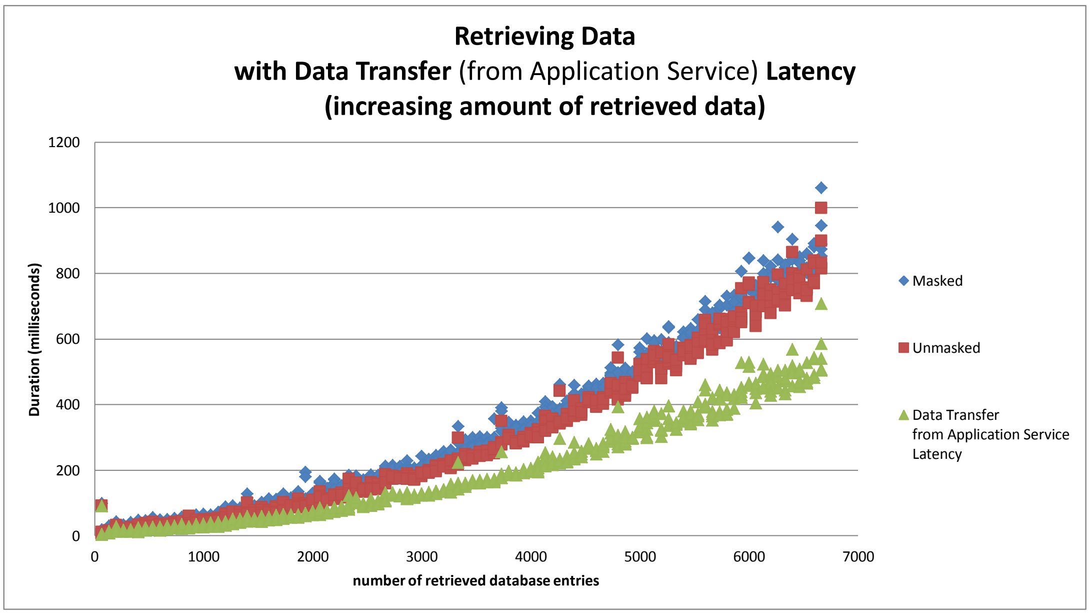


Figure 4.6: Retrieving Data with Data Transfer (from Application Service) Latency (increasing amount of retrieved data)

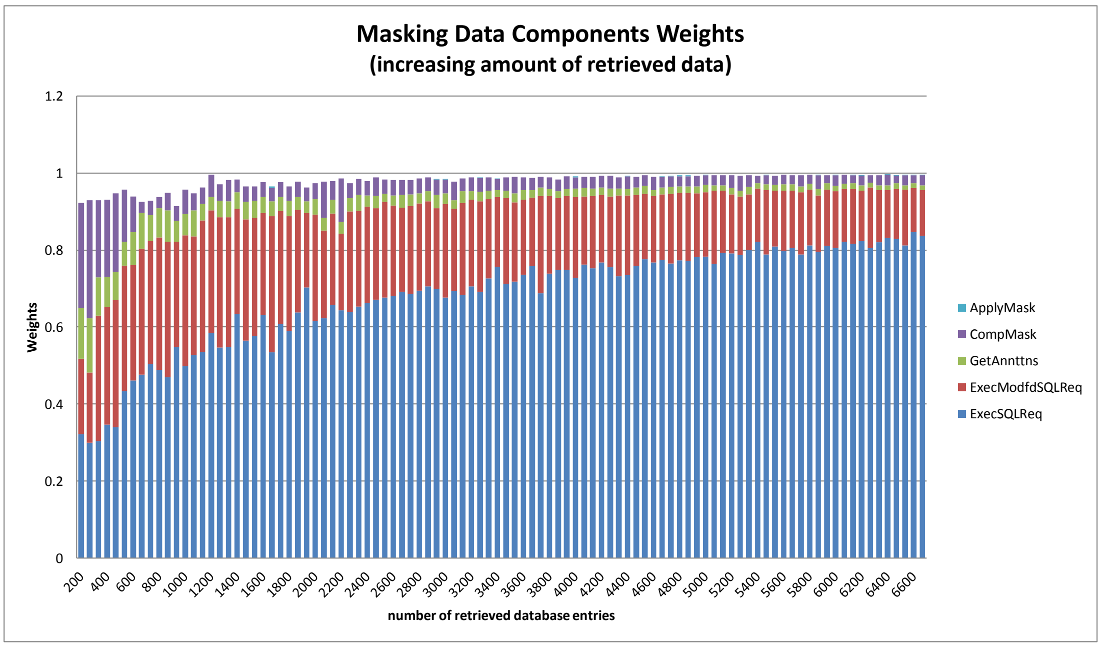


Figure 4.7: Masking Data Components Weights (increasing amount of retrieved data)

Figure 4.8 that the retrieval of masked data is more expensive than the retrieval of unmasked data, but that the performance of retrieving masked data is close to the performance of retrieving unmasked data.

More important is the fact that the duration of retrieving masked data from the database is influenced in a minimal way with the increase of the annotations attached to the data. This shows that the specific masking components (ExecModfdSQLReq, GetAnnttns, CompMask and ApplyMask) are efficient.

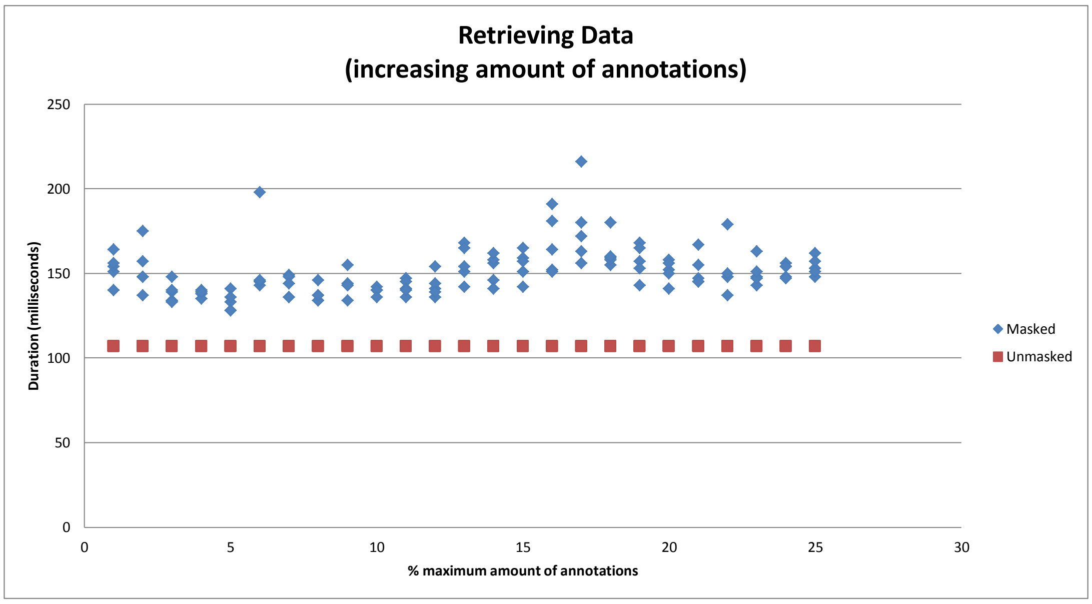


Figure 4.8: Retrieving Data (increasing amount of annotations)

When the data transfer (from application service) latency is added, it can be seen in Figure 4.9 that the red curve (retrieval of unmasked data) is getting closer to the blue curve (retrieval of masked data). This test shows that the user experience response times are even less affected in real world scenarios, when several latencies are part of users' experience.

4.4.2.4 Masking components weights test - annotations size

In this test the amount of annotations attached to the data was varied and we measured the weights of the components of the retrieval of masked data process. The plot depicted in Figure 4.10 shows that with the increase of the amount of annotations (up to 25% of the maximum number of annotations), the summed up weight of the specific masking components (ExecModfdSQLReq, GetAnnttns, CompMask and ApplyMask) stays constant. Each masking component has also the same behavior.

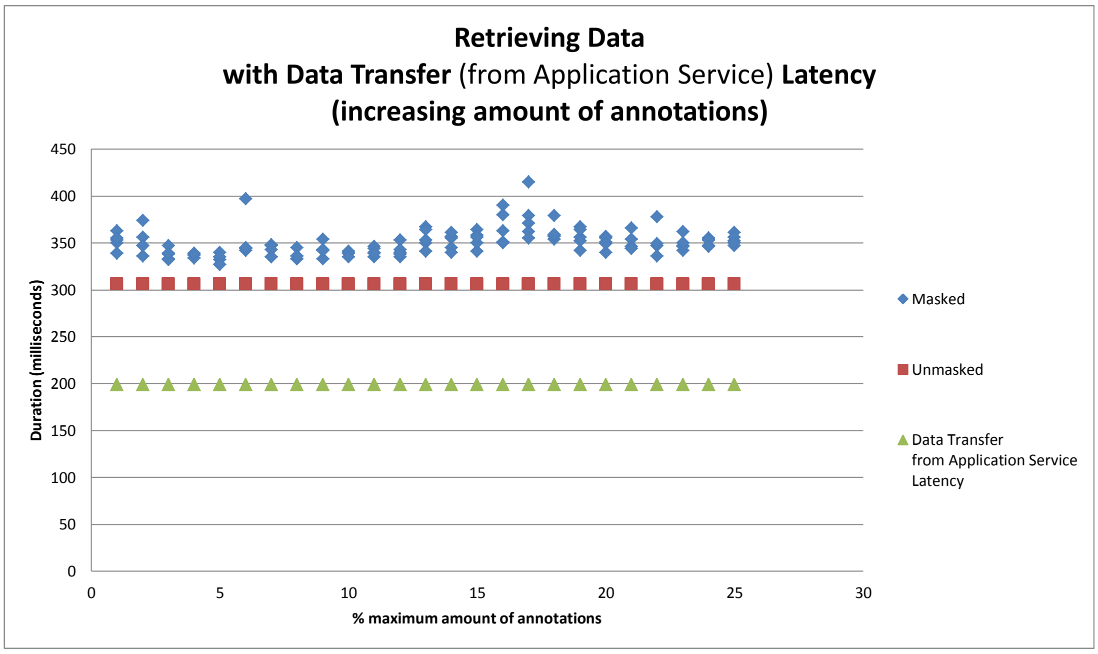


Figure 4.9: Retrieving Data with Data Transfer (from Application Service) Latency (increasing amount of annotations)

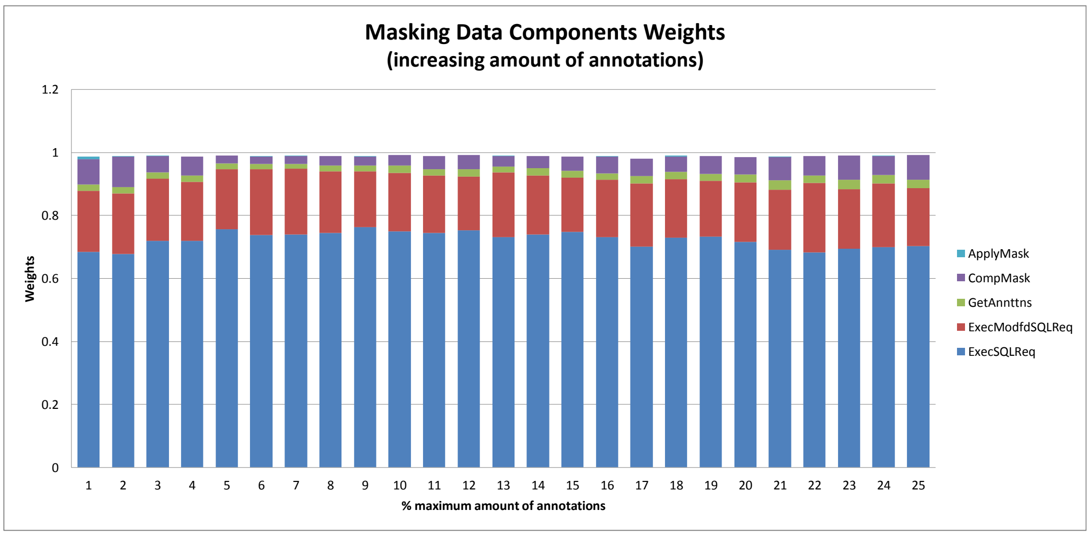
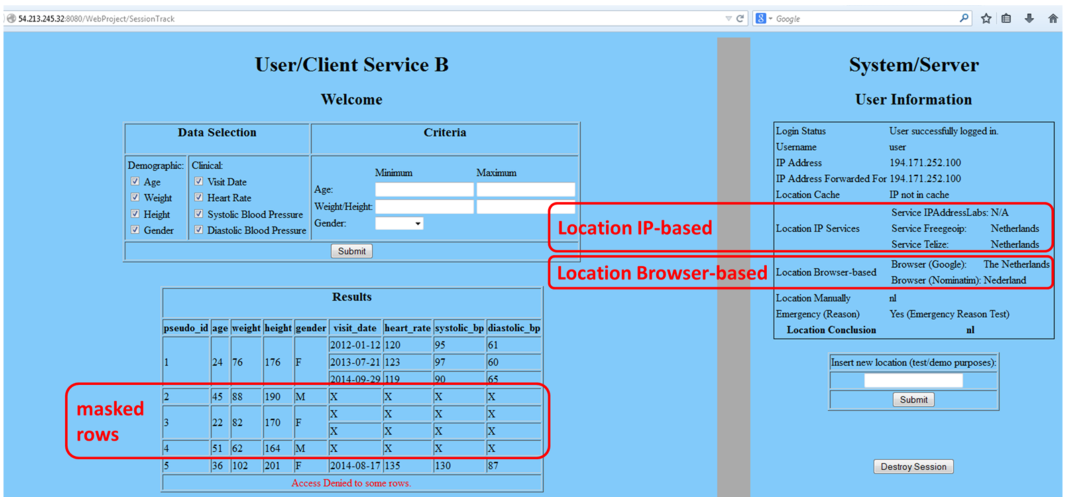


Figure 4.10: Masking Data Components Weights (increasing amount of annotations)

4.5 Results

Figure 4.11 depicts a print screen of an application service (web application) which is making use of geolocation in order to enforce data annotations and retrieve masked data. The application service contains the user/client side, but also information which is present on server side and about his location and his status of successfully being logged in.



User/Client Service B

Welcome

Data Selection		Criteria		
Demographic:	Clinical:		Minimum	Maximum
☑ Age	☑ Visit Date	Age:		
☑ Weight	☑ Heart Rate	Weight/Height:		
☑ Height	☑ Systolic Blood Pressure	Gender:		
☑ Gender	☑ Diastolic Blood Pressure			

Results

pseudo_id	age	weight	height	gender	visit_date	heart_rate	systolic_bp	diastolic_bp
1	24	76	176	F	2012-01-12	120	95	61
					2013-07-21	123	97	60
					2014-09-29	119	90	65
2	45	88	190	M	X	X	X	X
3	22	82	170	F	X	X	X	X
					X	X	X	X
4	51	62	164	M	X	X	X	X
5	36	102	201	F	2014-08-17	135	130	87

Access Denied to some rows.

System/Server

User Information

Login Status	User successfully logged in.	
Username	user	
IP Address	194.171.252.100	
IP Address Forwarded For	194.171.252.100	
Location Cache	IP not in cache	
Location IP Services	Service IPAddressLabs:	N/A
	Service Freegeoip:	Netherlands
	Service Telize:	Netherlands
Location Browser-based	Browser (Google):	The Netherlands
	Browser (Nominatim):	Nederland
Location Manually	nl	
Emergency (Reason)	Yes (Emergency Reason Test)	
Location Conclusion	**nl**	

Insert new location (test/demo purposes):

Figure 4.11: Geolocation and data sensitivity annotations proof of concept

Chapter 5

Conclusions

In this thesis we focused on the problem of unauthorized access to data shared across storage systems with different administrative domains. The risk of unauthorized access to the data is increased within collaborations because the collaborative parties may use different authorization mechanisms, which have to be synchronized. Furthermore supporting dynamic access control granularity levels in collaborations is also needed. However state of the art solutions for these problems have certain shortcomings. Therefore there is need in collaborations for developing interoperable, scalable and flexible authorization mechanisms to cope with conflict resolution and fine-grained access control.

A two-layer authorization architecture could be an answer to the research question on *How to manage and enforce policies across domains efficiently?*. The first layer is managing and enforcing the collaboration abstract policies, and the second layer uses domains specific policies. The abstract policies (agreed upon by all domains) use abstract resources and abstract operations, concepts shared by the collaborative parties. The two-layer architectural design facilitates overruling of the domains specific policies by the collaboration abstract policies. This is answering the research sub-question on *How to solve conflicts between collaboration access policies and domains specific access policies?* Conflict resolution becomes easier to perform within the domains. If the abstract policies permit or deny access to the data, then this decision is enforced. Otherwise, the policy enforcement point is enforcing the decision determined by the domain specific policies.

Besides being able to solve conflicts easier, the two-layer authorization architecture facilitates also efficient enforcement of abstract policies. This is done by abstracting the subject's request and evaluating the resulted abstract request against the abstract policies. This is more efficient than the traditional approach of evaluating subject's request against the domain specific policies because the number of abstract policies is smaller. We proved by complexity analysis and by experimental validation that abstracting the subject's request is not affecting the performance of the authorization system.

Additionally it was proven by complexity analysis that the administration of the policies is also more efficient. The administrators manage a smaller number of policies (only centrally) and manage the resources/operations mappings in the domains. The latter is again smaller than the possible number of extracted domain-specific policies.

An answer to the research question on *How to enhance the granularity level of the authorization to data based on its sensitivity status and using user attributes?* could be data sensitivity annotations. For the annotations we developed an approach to annotate rows and columns in a database with a sensitivity attribute or authorization policy. Like data, annotations are managed by a (new) platform service. Enforcement is left to application services via a policy component or SDK. Technically, it derives an annotations-query from the data-query. Experimentally the concept and performance for row-based and column-based annotations have been validated for practical use.

The results also carry a number of learnings especially with respect to obtain a performance level that is acceptable for data sensitivity annotations and data masking to deem practical and feasible. Most impactful factors here are efficient database interaction and optimization of data structures. In some cases NoSQL may prove useful, e.g. to retrieve annotations.

For geolocation control we developed a platform service that reliably determines the country of a remote user. It allows combination of location sources including IP-addresses, sensors as well as human input, which lends itself to be policy configurable to achieve the required level of trust and can also be combined with a break-the-glass option. It leverages location features present in clients such as mobiles and browsers a third party location services. We leveraged various sources for geolocation in order to minimize and deal better with false positives and false negatives. The proposed solution uses the geolocation as subject attribute in the authorization policy attached to data sensitivity annotations.

5.1 Future Work

Future work regarding the policy administration research question will include further analysis on how the placement of the policy decision point is affecting the level of trust between the collaborative domains. The discussion in section 3.2 is the basis for such an analysis. The minimization of the communication latency between a central policy decision and the domains policy enforcement points could also be addressed in future work.

Additionally we should validate the policy administration proposed solution against other XACML implementations, besides Sun XACML. Moreover we should leverage NoSQL for storage of resources/operations mappings.

Although the applied research provided many questions with an answer also new problems and questions appeared for the data sensitivity research questions. The open problems and the questions to be addressed in future work include:

- Newly inserted rows annotations. What annotations should be attached to the newly inserted rows and how? Should the solution leverage dynamic identification of data-to-annotate (e.g. where clauses)?
- Performance. Which parts of the algorithms involved in retrieval and enforcement of annotations could be improved? What are the implications of replacing the double execution of the original SQL request and obtain the required primary keys directly? Furthermore an SQL request analyser could be employed to output the tables involved in the SQL request. How performant is such an analyser.
- Cell-based annotations. How performant is the proposed solution when the annotations store contains cell-based annotations?
- Annotation algorithm. Does an efficient algorithm exist to lookup relevant annotations when the annotations are stored in a Key-Value store and the key is represented by the

data identifiers?

- Database changes. What updates have to be done when the database tables and columns change their names? Are these changes done automatically (database triggers leveraged) or by administrators, or semi-automatically assisted by administrators when needed?

The open problems and the questions to be addressed in future work for geolocation include:

- Accuracy: How to balance false positives and false negatives generated by IP-based geolocation services? “For IP-to-country database, some vendors claim to offer 98% to 99% accuracy although typical Ip2Country database accuracy is more like 95%.” [36]
- Trust: How much does the server trust in the location received from the client? Is there need for tamper proofing location-related components on client side?
- Decision: How is it decided if the geolocation determination action is successful or not? What happens if the information from the server and the client contradict?
- Integration: How to support minimal developer integration effort with protocols and API while also supporting a good user experience for user interaction parts?

Bibliography

[1] Florian Kelbert and Alexander Pretschner. Towards a Policy Enforcement Infrastructure for Distributed Usage Control. In *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, SACMAT '12, pages 119–122, New York, NY, USA, 2012. ACM. xi, 8

[2] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence. AAA Authorization Framework. RFC 2904, RFC Editor, August 2000. 6

[3] R. L. Kissel. NIST Interagency Report (NISTIR) 7298, Glossary of Key Information Security Terms. Technical report, April 2007. 6, 7

[4] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The NIST Model for Role-based Access Control: Towards a Unified Standard. In *Proceedings of the Fifth ACM Workshop on Role-based Access Control*, RBAC '00, pages 47–63, New York, NY, USA, 2000. ACM. 7

[5] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone. Guide to Attribute Based Access Control (ABAC) Definition and Considerations. NIST Special Publication 800-162, January 2014. 7, 9

[6] N. Sahavechaphan, S. U-ruekolan, N. Harnsamut, J. Phengsuwan, and K. Aroonrua. An efficient technique for aspect-based EHR access policy administration on ABAC. In *ICT and Knowledge Engineering (ICT Knowledge Engineering), 2011 9th International Conference on*, pages 27–33, Jan 2012. 7, 9, 12

[7] R.P. Koster, M. Asim, and M. Petkovic. End-to-End Security for Personal Telehealth. In *MIE 2011; 23rd Int. Congress Eur. Federation Medical Informatics (Oslo, Norway)*, Septmeber 2011. 7

[8] M. Asim, R.P. Koster, M. Petkovic, and M. Rosner. eConsent Management and Enforcement in Personal Telehealth. In Helmut Reimer, Norbert Pohlmann, and Wolfgang Schneider, editors, *ISSE 2012 Securing Electronic Business Processes*, pages 209–216. Springer Fachmedien Wiesbaden, 2012. 7

[9] S. Betge-Brezetz, G.-B. Kamga, M.-P. Dupont, and A Guesmi. End-to-end privacy policy enforcement in cloud infrastructure. In *Cloud Networking (CloudNet), 2013 IEEE 2nd International Conference on*, pages 25–32, Nov 2013. 8

[10] Bo Tang, Qi Li, and R. Sandhu. A multi-tenant RBAC model for collaborative cloud services. In *Privacy, Security and Trust (PST), 2013 Eleventh Annual International Conference on*, pages 229–238, July 2013. 8

[11] W. Han, Z. Fang, L.T. Yang, G. Pan, and Z. Wu. Collaborative Policy Administration. *Parallel and Distributed Systems, IEEE Transactions on*, 25(2):498–507, Feb 2014. 8

[12] S. Alshehri and R.K. Raj. Secure Access Control for Health Information Sharing Systems. In *Healthcare Informatics (ICHI), 2013 IEEE International Conference on*, pages 277–286, Sept 2013. 9

[13] A. Kim, J. Luo, and M. Kang. Security Ontology for Annotating Resources. In *Proceedings of the 2005 OTM Confederated International Conference on On the Move to Meaningful Internet Systems: CoopIS, COA, and ODBASE - Volume Part II*, OTM'05, pages 1483–1499, Berlin, Heidelberg, 2005. Springer-Verlag. 9

[14] Row-security - PostgreSQL wiki. `http://wiki.postgresql.org/wiki/RLS`. Accessed: April 7, 2014. 10

[15] Implementing row level security in MySQL - SQL Maestro Group. `http://www.sqlmaestro.com/resources/all/row_level_security_mysql/`. Accessed: April 7, 2014. 10

[16] Using Oracle Virtual Private Database to Control Data Access. `http://docs.oracle.com/cd/E11882_01/network.112/e16543/vpd.htm`. Accessed: April 7, 2014. 10

[17] Enforcing Row-Level Security with Oracle Label Security. `http://docs.oracle.com/cd/E16655_01/server.121/e17609/tdpsg_ols.htm`. Accessed: April 7, 2014. 10

[18] Implementing Row- and Cell-Level Security in Classified Databases Using SQL Server 2005. `http://technet.microsoft.com/en-US/library/cc966395.aspx`. Accessed: April 7, 2014. 10

[19] Apache Accumulo User Manual Version 1.5. `https://accumulo.apache.org/1.5/accumulo_user_manual.html`. Accessed: April 7, 2014. 10

[20] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.*, 26(2):4:1–4:26, June 2008. 10

[21] NoSQL Databases. `http://www.christof-strauch.de/nosqldbs.pdf`. Accessed: July 17, 2014. 10

[22] Peyman Nasirifard, Vassilios Peristeras, and Stefan Decker. Annotation-based Access Control for Collaborative Information Spaces. *Comput. Hum. Behav.*, 27(4):1352–1364, July 2011. 10, 12

[23] N.A.T. Tran and T.K. Dang. A Novel Approach to Fine-grained Content-based Access Control for Video Databases. In *Database and Expert Systems Applications, 2007. DEXA '07. 18th International Workshop on*, pages 334–338, Sept 2007. 10

[24] Xueli Li, N.A Naeem, and B. Kemme. Fine-granularity access control in 3-tier laboratory information systems. In *Database Engineering and Application Symposium, 2005. IDEAS 2005. 9th International*, pages 391–397, July 2005. 10

[25] A. Maamir, A. Fellah, and L.A. Salem. Fine Granularity Access Rights for Information Flow Control in Object Oriented Systems. In *Information Security and Assurance, 2008. ISA 2008. International Conference on*, pages 122–128, April 2008. 11

[26] IP Address Geolocation to Identify Website Visitor's Geographical Location. `http://www.ip2location.com/`. Accessed: April 7, 2014. 11

[27] IPInfoDB — Free IP Address Geolocation Tools. `http://www.ipinfodb.com/`. Accessed: April 7, 2014. 11

[28] S. Merigeault, M. Batariere, and J.N. Patillon. Data fusion based on neural network for the mobile subscriber location. In *Vehicular Technology Conference, 2000. IEEE-VTS Fall VTC 2000. 52nd*, volume 2, pages 536–541 vol.2, 2000. 11

[29] J. M. Zagami, S. A. Parl, J. J. Bussgang, and K. D. Melillo. Providing Universal Location Services Using a Wireless E911 Location Network. *Comm. Mag.*, 36(4):66–71, April 1998. 11

[30] A. Giordano, M. Chan, and H. Habal. A novel location-based service and architecture. In *Personal, Indoor and Mobile Radio Communications, 1995. PIMRC'95. Wireless: Merging onto the Information Superhighway., Sixth IEEE International Symposium on*, volume 2, pages 853–857 vol.2, Sep 1995. 11

[31] Location-Based Authorization. `http://www-users.cs.umn.edu/~shekhar/research/marcus.pdf`. Accessed: April 7, 2014. 11

[32] Mozilla Firefox Web Browser Geolocation in Firefox Mozilla. `https://www.mozilla.org/en-US/firefox/geolocation/`. Accessed: April 7, 2014. 11

[33] Geolocation - Dive Into HTML5. `http://diveintohtml5.info/geolocation.html`. Accessed: April 7, 2014. 11

[34] Design of Identity Management, Authentication and Authorization Infrastructure. `http://homes.esat.kuleuven.ac.be/~decockd/tas3/final.deliverables/pm36/TAS3_D07p1_IDM-Authn-Authz_V3p0.pdf`. Accessed: July 17, 2014. 12

[35] S. Marouf, M. Shehab, A Squicciarini, and S. Sundareswaran. Adaptive reordering and clustering-based framework for efficient xacml policy evaluation. *Services Computing, IEEE Transactions on*, 4(4):300–313, Oct 2011. 22

[36] IPLocation. `http://www.iplocation.net/`. Accessed: May 6, 2014. 45

Appendix A

Use Case: Translational Research (Movember)

A.1 Description

The Movember Global Action Plan 3 (GAP3) on Active Surveillance (AS) of prostate cancer (PC) includes the activity of 14 areas (Research Institutes and University Medical Centres) in the 5 Movember regions (Australasia, Europe, UK, Canada, USA) to construct a sustainable worldwide database for clinical, marker-related, and imaging data for scientific analyses and improvement of clinical guidelines.

Movember GAP3 aims at:

1. creating a global consensus on selection and monitoring of men with low risk cancer;
2. managing a worldwide platform with information and guidelines on AS as acknowledged treatment option for PC;
3. reduction of number of men switching towards active therapy within one year after start of the AS protocol;

Movember is a translational research use case. The translational medical research is an emerging field of work, which aims to bridge the gap between basic medical science research and clinical research/patient care. Figure A.1 shows high-level translation research data flows.

Given that Movember must deal with very sensitive data, accessed from different organizations (Research Institutes and University Medical Centres) and by different users (biostatisticians, technicians or radiologists) in a distributed system, it is of paramount importance to ensure proper user authentication and authorization, data protection, consent management and enforcement and information trustworthiness (reliability).

Figure A.2 shows the use case flow. The Use Case Flow also emphasises in several places the need for user authentication and authorization:

- In order to comply with the legislation, in translational research patient permissions to anonymise/pseudonymise and to use their data must be properly regulated (proper patient digital consent management).

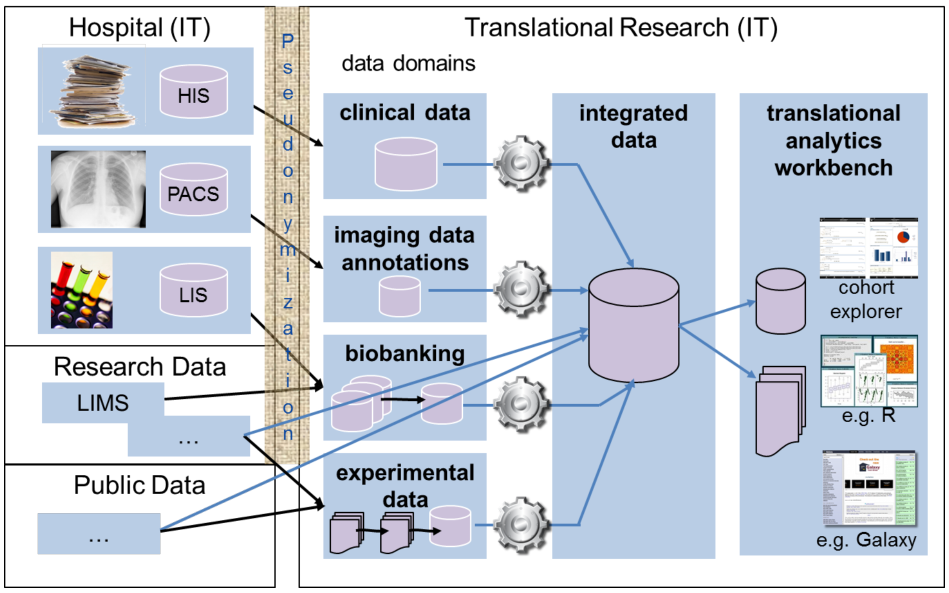


Figure A.1: High level translation research data flows

- The Demilitarised Zone (DMZ, Figure A.2: Use Case Flow) is a temporary landing zone for the anonymised data which comes from the organizations (Research Institutes and University Medical Centres). The data uploaded in the DMZ by each organization is anonymised before the upload by the organization itself. The DMZ must be separated in landing zones, which are mapped to the organizations that are uploading anonymised data. The DMZ must have authentication and authorization in place in order to regulate the access to the landing zones and protect them. Only the organizations can upload data to their DMZ corresponding landing zone. The access control policy of the DMZ must forbid the organizations to have access to any other part of the DMZ, except their DMZ landing zone. The separation of the DMZ in landing zones facilitates respecting the legislation of all the 5 Movember regions (Australasia, Europe, UK, Canada, USA).

- The access to the DMZ data is different when the correct data anonymization checking step (step 2, Figure A.2: Use Case Flow) is realised. Authentication and authorization must be present again in this situation. Only the data stewards belonging to the organization that is hosting: “Globally accessible shared translational data” must be able to do the checking step.

- Finally, access to sensitive anonymised medical information (step 3 in Figure A.2, e.g. PACS records) is usually regulated by authentication and access control policies, which specify which parties and under what conditions may access information. If the information is confined within a single, trusted system, policy enforcement can be achieved using traditional enforcement mechanisms. When information needs to be disclosed across different organizational and jurisdictional domains (e.g. different Research Institutes and University Medical Centres that host biostatisticians and researchers), however, guaranteeing policy enforcement becomes more challenging. Therefore attribute-based policy enforcement techniques will play an important role here.

In order to put in practice this use case, security and privacy related criteria that have to be satisfied are:

- The research institutions are able only to write data into their DMZ partitions.
- The research institutions are not able to read data from DMZ.
- Only the Quality Checker (person or tool) is able to read data from DMZ.
- The Quality Checker is not able to write data in the DMZ.
- If the Quality Checker's test fails, then this means that the incorrect anonymised data is rejected and the Quality Checker communicates this to the owner of the incorrect anonymised data.
- When the research institution is informed that its uploaded data is not correctly anonymised, the research institution can:
  - re-anonymise its data and re-upload it in its DMZ partition. Now the Quality Checker will verify the new updated data.
  - not do anything. In this case none of its anonymised data goes to "Globally accessible shared translational data".
- Not all the members from an organization will necessarily have the same access privileges to "Globally accessible shared translation research data".
- Access to the "Globally accessible shared translational research data" is limited to those who have the knowledge to make statistical analysis.

A.2 Actors

The list of actors is:

- Patients. The patients give consent for their clinical data to be used in the Movember statistical research;
- Project Leaders and IT Experts from the organizations (Research Institutes and University Medical Centres). These actors are uploading anonymised data to DMZ (step1, Figure A.2: Use Case Flow);
- Data Stewards. The data stewards are responsible for the execution of steps in 2 (2.Accept, 2.Reject) from the Use Case Flow (Figure A.2). They belong to the company which is hosting the "Globally accessible shared translational data";
- Clinical Researchers at Research Institutes and University Medical Centres. These clinical researchers are: biostatisticians, technicians and radiologists. They are using the data (e.g. making statistics) in step 3 of Use Case Flow (Figure A.2);

A.3 Trigger

There is need to:

- address one of the critical issues facing men diagnosed with prostate cancer today – whether to treat low risk prostate cancer or monitor it via active surveillance;
- accelerate health outcomes for men living with prostate and testicular cancer;

Translational Research (Movember) is breaking down barriers to research, funding worldwide collaboration, uncovering the best talent and bringing them together to work as a team. This is done by collecting data and by undertaking clinical research.

A.4 Preconditions

To Be Decided

A.5 Postconditions

To Be Decided

A.6 Priority

The priority of this use case is high as the use case is being implemented in real practice and if security is not properly regulated, given the sensitivity of health information, this might be a big barrier for the acceptance of the use case.

A.7 Frequency of Use

The frequency of use is determined by the frequency of anonymised data uploads to DMZ (step 1, Figure A.2: Use Case Flow). The data uploads happen when an organization (Research Institute or University Medical Centre) wants to upload data in the DMZ. As estimation, this operation is expected to happen 3-4 times per year for each organization. Each time the process shown in Figure A.2 will be executed.

The frequency of use may be affected also by the rejection of the anonymised data. In this case the upload of the data to the DMZ may happen multiple times because of the incorrect data anonymization. The checker (step2, Figure A.2: Use Case Flow) will delete the incorrect anonymised uploaded data and will ask the organization to re-execute the anonymised data upload (step 1, Figure A.2: Use Case Flow).

A.8 Use Case Flow (Normal Course + Alternative Courses of Events)

The flow of the AU2EU Translational Research Use Case is shown in Figure A.2 and is compound of the next steps:

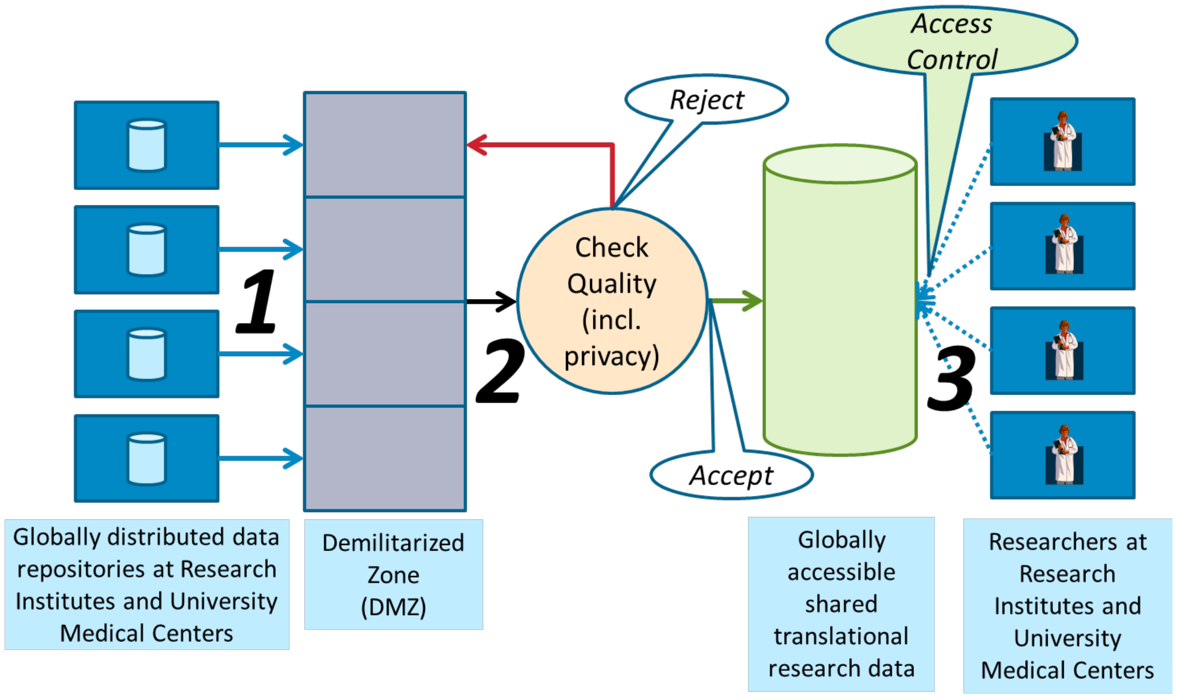

Access Control
Reject
1
2
3
Check Quality (incl. privacy)
Accept
Globally distributed data repositories at Research Institutes and University Medical Centers
Demilitarized Zone (DMZ)
Globally accessible shared translational research data
Researchers at Research Institutes and University Medical Centers

Figure A.2: Use Case Flow

1. **The organizations (Research Institutes and the University Medical Centres) upload the anonymised data in the Demilitarised Zone (DMZ).** The data uploaded in the DMZ by each organization is anonymised before the upload by the organization itself. In order for this upload to become possible the DMZ has to be proven that is secure:

   - It has to be proven that the landing zones are separated and that the organizations have access only to their landing zone.
   - It would be preferable for the organizations to have only the right of uploading the anonymised data (only one way, from the organization to their DMZ landing zone).
   - The DMZ must be a solution which respects the legislation of all the places that the data is held and made available.
   - The DMZ must not be reachable from outside except for the organizations (Research Institutions and University Medical Centres) in this step 1. For step 2 the DMZ must be reachable also by the data stewards who are doing the checking.
   - The question "Who is responsible for the DMZ?" must also be solved. This depends also on the DMZ requirements (mentioned above).
   - DMZ is a temporary storage of the anonymised patient data and the data in the DMZ is consumed by the checker in step 2.

2. **The uploaded data is checked if it is properly anonymised.** The checking is done by data stewards. The data stewards are able to *read* and *delete* the data from

the DMZ. They are also in charge of harmonizing the data. Depending of the result of the checking two things can happen; the upload may be accepted or rejected:

- **Accept.** If the uploaded data is correctly anonymised, then the data is moved in "Globally accessible shared translational research data". Now the data can be used by biostatisticians in step 3.
- **Reject**. If the uploaded data is not correctly anonymised, then the data is removed from the DMZ and the organization which uploaded the data is asked to upload the correctly anonymised data, step 1.

3. **The data is consumed (e.g. read, export) by the organizations actors: biostatisticians, technicians and radiologists.** Here user authentication and authorization must be present in order for the data protection to be provided.

   - Not all the members from an organization will necessarily have the same access privileges.
   - Access to the "Globally accessible shared translation research data" is limited to those who have the knowledge to make statistical analysis. This is meant to prevent that people derive conclusions on the large data set and derive bad conclusions.
   - The access control policy will be most probably role-based (specialization of attribute-based access control policy)
   - The system may need to *export* the data as specialised statistical tools may be present only at the biostatisticians' organization.

The main flow of the use case is: 1 → 2.Accept → 3.

A.9 Exceptions

Error conditions can be:

- The anonymised data passed the checker, but it is not actually anonymised correctly (e.g. birthday is visible in the data). This type of error can be observed by humans, not automatically. The solution in this case is a process solution. The error is reported to a data steward. The data steward removes the incorrectly anonymised data and asks the data owner organization to re-upload the correctly anonymised data.

A.10 Includes

N/A (Not Applicable)

A.11 Special Requirements (Non-Functionals)

The organizations must have the patients' consent in order to use their data in translation research.

The DMZ part of the data upload from the organizations (Research Institutes and University Medical Centres) to the “Globally accessible shared translational research data” must affect the least the total duration of the data upload. (performance requirement)

The DMZ must respect all data protection regulations and laws from all 5 Movember regions (Australasia, Europe, UK, Canada, USA).

A.12 Assumptions

A legislative solution of putting all the data from all 5 Movember regions (Australasia, Europe, UK, Canada, USA) together is achievable.

A.13 Notes and Issues

To Be Decided

Appendix B

Use Case: Collaborative PACS

B.1 Description

A picture archiving and communication system (PACS) is a medical imaging technology that provides storage of, and convenient access to, images from multiple modalities. PACS has four main uses, namely: (i) hard copy replacement; (ii) remote access that provides capabilities of off-site viewing and reporting enabling practitioners in different physical locations to access the same information simultaneously; (iii) an electronic image integration platform that provides interfacing with other medical automation systems such as Electronic Medical Record (EMR) systems; and (iv) radiology workflow management.

Given that PACS must deal with very sensitive data, accessed from different organizations (hospitals) and by different users in a distributed system, it is of paramount importance to ensure proper user **authentication** and **authorization**. In order to comply with legislation, in medical systems like PACS patient permissions to access data must be properly regulated (proper patient digital **consent** management). Such a user-centric, **federated** approach contrasts to the "back-end attribute exchange" model, which typically involves transmission of user information without involvement or consent by the user, and the use of common identifiers across domains. Furthermore remote access by healthcare providers from their own mobile devices to PACS is also emerging. Assurance of **trustworthiness** of the mobile device and attribute claims play an important role here as well. Finally, access to sensitive medical information as PACS records is usually regulated by access control policies, which specify which parties and under what conditions may access information. If the information is confined within a single, trusted system, policy enforcement can be achieved using traditional enforcement mechanisms.

At this moment PACS is a standalone product which is used in hospitals by installing the PACS solution in the hospital including image database and the necessary software. The workflow / administrative use scenarios are:

1. Displays and provide interaction with worklists for radiologists;
2. Assign work to one or more radiologists;
3. Communicate to other clinicians via chat/messaging functionality (only in system boundary);
4. Perform administrative tasks to manage the system.

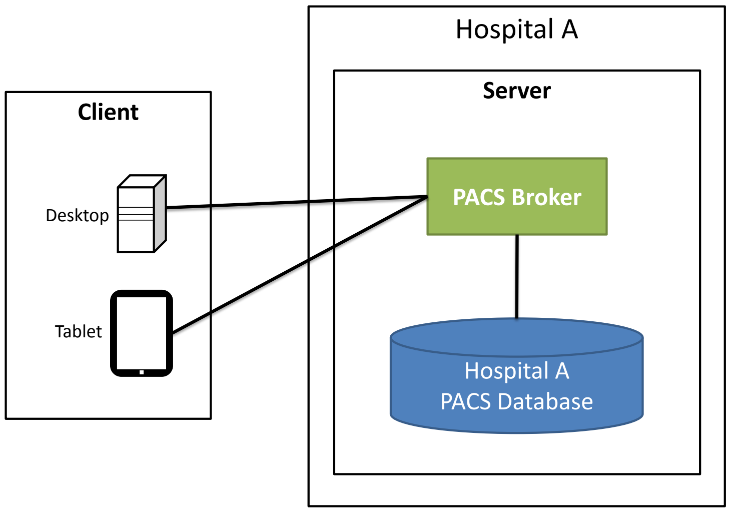


Figure B.1: Single Hospital PACS Architecture

In Figure B.1: Architecture the system shows that the PACS system is a Client-Server architecture. This architecture involves a user (Client) which can be outside the hospital and can use a Desktop or a Tablet. In this architecture the hospital has a Server which contains a main building block: PACS Broker. PACS Broker is the frontend component of the PACS system and it is handling the Client-Server connections. So PACS Broker is also handling the Client authentication and authorization, which are the main security mechanisms contained in this architecture. The PACS Database contains patient data (demographic and medical data) and audit data. Authorization is also present from the PACS Broker to PACS Database. Consent though is not applicable as only the hospital is dealing with the consent of the patients and in consequence the management and enforcement of the patients consent is not taken in account during authorization in Figure B.1: Architecture.

More challenging from authentication and authorization, data protection, consent management and enforcement and information trustworthiness points of view is the scenario when information needs to be disclosed across different organizational and jurisdictional domains (e.g. different hospitals that host practitioners involved in teleradiology). The logical architecture of collaborative PACS is summarised in Figure B.2: Collaborative PACS Architecture.

Figure B.2: Collaborative PACS Architecture shows two hospitals which are sharing images. The need for sharing is given by the collaboration present in the Tumour Board part of the picture. A Tumour Board is formed of people (e.g.: radiologist, cardiologists) from different hospitals/organizations. The Tumour Board in the picture uses the PACS sharing system by accessing images from two hospitals in their reviews and discussions regarding the medical condition and treatment options of a patient.

In this scenario two hospitals which are sharing their images are present. In one case the images are stored in the hospital (e.g. Hospital B). In the other case the images are stored using a Cloud Service (e.g. Hospital A). This makes this use case more challenging from the needed security mechanisms point of view. As this picture is a logical view of Collaborative PACS, instead of/next to the PACS Brokers from all the hospitals a central Broker which

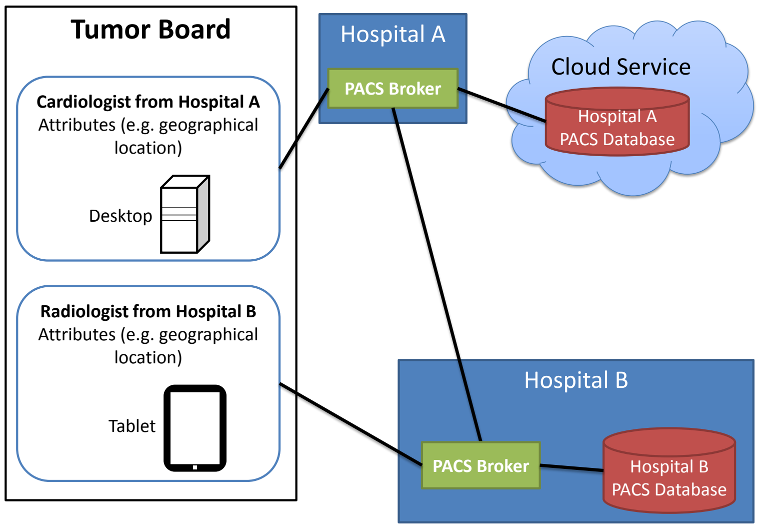

Tumor Board
Cardiologist from Hospital A
Attributes (e.g. geographical location)
Desktop
Radiologist from Hospital B
Attributes (e.g. geographical location)
Tablet
Hospital A
PACS Broker
Cloud Service
Hospital A PACS Database
Hospital B
PACS Broker
Hospital B PACS Database

Figure B.2: Collaborative PACS Architecture

has no affiliation to any hospital might be present in order to intermediate the users requests. The users can use desktops or tablets for getting access to the patients data.

In the sharing system the consent management becomes suddenly very important. Inside a hospital the consent was collected and used only at the moment of patient giving consent for their personal data to be used inside that hospital. Once the sharing and collaboration between hospitals become part of the picture, the patients consent gets highly involved in the policy enforcement and consequently in the authentication and authorization of user across organization even from different countries.

In this scenario security mechanisms like SSO (Single Sign-On) might be present. In this case federation of identities from the hospitals might be taken in consideration in the authentication solution.

It is desired that the authorization policies take in consideration the location of the user. For instance, a French physician using his mobile device to access images while he is on holiday in Peru should be restricted. Moreover, the granularity of the authorization has to be at the level database tables fields. The authorization policies should be able to specify for the database tables fields: sensitivity of the field, level of encryption required, permitted locations for storage. This implies database segmentation and different sensitivity levels. Therefore attribute-based policy enforcement techniques might play an important role here.

Furthermore the data can reside partially in the cloud; partially in the hospital. This leads the Collaborative PACS system to a hybrid cloud solution. The system should support policies which specify geographical boundaries for the storage and traffic of data.

In order to put in practice this use case, security and privacy related criteria that have to be satisfied are:

- Users should be able to login using the credentials from their hospital.
- The patients consent must be collected and enforced before any patient data lands in the PACS database.

- Patients consent must be enforced every time the patient data gets accessed.
- The geographical localisation of the user who wants to get access to data must be used in authorization enforcement.
- The policies must be stored in or next to the PACS database.
- Modifications of the policies must be reflected immediately (5 seconds) in all policy enforcement actions from the whole Collaborative PACS system.
- Not all the users belonging to the same hospital will necessarily have the same access privileges to the patients data.

B.2 Actors

The list of actors is:

- Advanced users: radiologist, technician, clinical researcher, general practitioner, perhaps the patient and other roles;
- System Administrators (hospital employee, or a role of the service engineer);
- Service Engineers (PACS vendor employee, but also service engineers for third party products, e.g. the hardware).

B.3 Trigger

There is need to make possible and then accelerate collaboration between healthcare institutions involving picture archiving domain. The acceleration will make possible for patients to receive collaborative treatments. Research can be another beneficiary.

B.4 Preconditions

To Be Decided

B.5 Postconditions

To Be Decided

B.6 Priority

MEDIUM
This use case aims at (no so distant) future use. It supports trends like collaboration in expert panels and cloud-based PACS services. From innovation point of view, it should be addressed now.

B.7 Frequency of Use

The frequency of use depends on the amount of collaboration between hospitals and especially on the hospitals to which the patients are going or on the specialised tools from within the hospitals.

For example an image processing tool might be available only in hospital H. Then whenever a patient P (from any collaborative hospital CH) needs in his treatment the tool from H then hospital H has to be able to retrieve Ps needed image in order to be able to process it. In this case, as the specialised tool is present only in H, the frequency of collaborative PACS is big.

B.8 Use Case Flow (Normal Course + Alternative Courses of Events)

The use case flow for the collaborative scenario implies more hospitals, each hospital with its PACS Broker and PACS database. This use case presents how a patient data request is dealt with in Collaborative PACS. The flow of the Collaborative PACS shown in Figure B.3: Use Case Flow and is compound of the next steps (only red links):

1. Hospital B radiologist connects to the Hospital B PACS Broker using a tablet.
2. Hospital B radiologist authenticates to the Hospital B PACS Broker.
3. Hospital B radiologist requests patient data which resides in Hospital A PACS database.
4. Hospital B PACS Broker redirects the radiologist to Hospital A PACS Broker.
5. Hospital A PACS Broker puts together the policies from the Policies Logical Storage and the attributes of the radiologist and decides if to disclose the requested patient data or not.
6. Hospital A PACS Broker enforces the authorization rules by disclosing or rejecting radiologists request and sending the answer to Hospital B PACS Broker.
7. Hospital B PACS Broker forwards the answer to Hospital B radiologist.

The black links are not part of the use case and they show only that other connections are present in the system. For example Hospital B PACS Broker has a connection with Policies Logical Storage component. This connection is needed when a request for an image in Hospital B PACS Database has to be honored. In this case Hospital B PACS Broker follows the policies (policy enforcement) from the Policies Logical Storage.

The steps in the above use case flow may be slightly different and this depends on where the broker functionality is located. Also the Policy Logical Storage in reality can be at different places, as long as it is associated with the data.

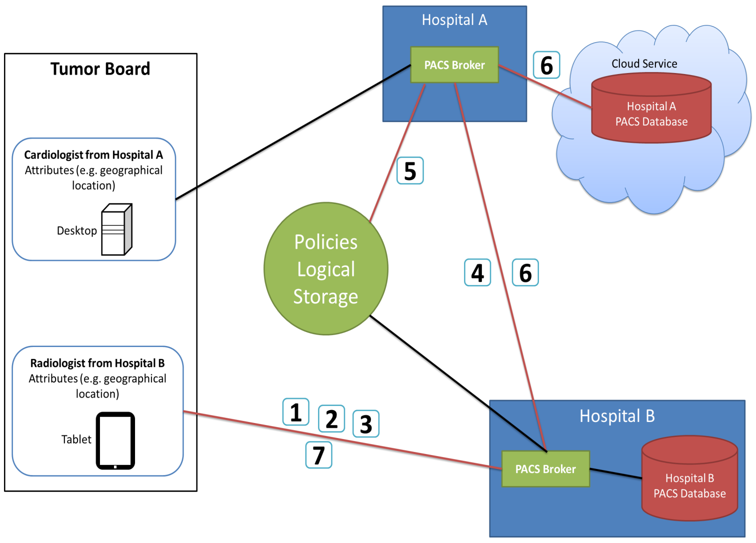


Figure B.3: Use Case Flow

B.9 Exceptions

Error conditions can be:

- An error is when a broker uses an old policy (which was updated in the meantime).

B.10 Includes

N/A (Not Applicable)

B.11 Special Requirements (Non-Functionals)

To be determined.

As a guideline, in transitioning from to the collaborative system, the the user should not notice more than a 10% decrease in system responsiveness.

B.12 Assumptions

Like in the translational research use case, depending on the size of the collaboration (national or international), legislative problems should not be encountered (e.g. a solution for sharing patient consent has to be possible).

B.13 Notes and Issues

To Be Decided

Appendix C

Requirements: Translational Research (Movember)

Glossary

Acronym	Full name/Description
ABC4Trust	An EU-funded research and development project advancing the federation and interchangeability of technologies supporting trustworthy and at the same time privacy-preserving Attribute-based Credentials.
TAS3	Trusted Architecture for Securely Shared Services. A project aiming to have a European-wide impact on services based upon personal information, which is typically generated over a human lifetime and therefore is collected and stored at distributed locations and used in a multitude of business processes.
TDL	Trust in Digital Life. An FP7 funded research community for the research and development of trustworthy ICT solutions.
Trust-PDP	Trust Policy Decision Point. Responsible for coordinating and combining trust services for nested policies.

C.1 Description

C.1.1 Introduction and background

The Movember Global Action Plan 3 on active surveillance of prostate cancer aims to construct a sustainable worldwide database for clinical, marker-related, and imaging data for scientific analyses and improvement of clinical guidelines.

Given that the Movember project must deal with very sensitive data, accessed from different organisations (Research Institutes and University Medical Centres) and by different users (biostatisticians, technicians or radiologists) in a distributed system, it is of paramount importance to ensure proper user authentication and authorisation, data protection, consent management and enforcement and information trustworthiness (reliability).

C.2 Functional Requirements

A part of the identified requirements below are inspired by TAS3, ABC4Trust and TDL security requirements. These requirements are signalled by references to the inspiring requirements. Such a reference has the form

$$TAS3|ABC4Trust|TDL, requirementnumber$$

The requirement number is the one from AU2EU State-of-the-Art Requirements document.

C.2.1 Authentication

- Authentication method

  1. The system MUST support authentication, for example, using a username and password pair as authentication information.
  2. If a password is used, then it SHOULD comply with a set of constraints in order to increase its strength. Such a set of constraints MAY be:
     (a) The password MUST contain numbers.
     (b) The password MUST contain special characters.
     (c) The password MUST have a length greater than 10 characters.
  3. The system MAY provide (e.g. administrator) authentication from specific whitelisted IP addresses only. (device authenticity)
  4. Users MUST be able to authenticate to services using their existing authentication credentials without needing to be issued with new authentication credentials (whatever form those credentials may take). [TAS3, 27]
  5. Users MUST be enrolled with an Identity Provider in order to use the services of its mutually trusted Service Providers. [TAS3, 28]
  6. Users SHOULD be allowed to enrol with multiple Identity Providers under different unrelated identities. [TAS3, 29]
  7. In case of strong authentication, a hardware card which generates presentation tokens based on privacy attribute based credentials solution MAY be used. [ABC4Trust,1]
  8. Privacy attribute based credentials MUST be able to be bound to the hardware card and/or to the user. [ABC4Trust,3]
  9. The user MUST not be able to manipulate the presentation tokens or the privacy attribute based credentials without damaging their integrity. [ABC4Trust, 4]
  10. Privacy attribute based credentials MUST be stored on the hardware card. [ABC4Trust, 5]
  11. A user MUST be able to read all contents of his/her hardware token (except the private key contained in the card). [ABC4Trust, 7]

12. If the hardware token contains a users secret, the user MUST not be able to read the users secret. [ABC4Trust, 7]
13. The system MUST prevent the users from generating tokens from attributes not certified by their own privacy attribute-based credentials. [ABC4Trust, 17]
14. A replay of the same presentation token MUST not be allowed by the system. [ABC4Trust, 19]
15. The audit logs MUST never reveal the values of non-public keys and secrets. [ABC4Trust, 21]

- User accounts

16. A user SHOULD be forced to change his/her password at least once in 6 months.
17. All user accounts SHOULD have expiration dates.
18. The administrator MUST be able to delete or deactivate user accounts which have reached their expiration dates.
19. The systems administrator MAY check once or twice per year for expiring user accounts.
20. People SHOULD be able to use their existing accounts when taking part in processes of organisations they did not have contact with before. [TAS3, 1]
21. The system SHOULD minimise the amount of data collected from the users. [TDL, 17]

- User identification

22. The system MUST uniquely identify each user using an email address.

C.2.2 Authorisation

- Access Control Policy

1. The systems access control policy MUST support role-based access control (individuals are assigned to roles and roles are assigned different permissions).
2. The systems access control policy MAY support the concept of groups of users.
3. Service Providers MUST be able to set the policies (authentication, authorisation and trust) that control access to the resources under their control. [TAS3, 30]
4. Users SHOULD be able to set the policies that control access to their personal identifying information (PII), and this policy SHOULD stay with their PII regardless of where this PII is subsequently held. [TAS3, 31]
5. Administrators MUST be able to set the policies that control legal access to PII. [TAS3, 32]
6. It MUST be possible to specify separation of duty constraints, i.e. that some tasks MAY not be performed by the same person. [TAS3, 5]
7. It MUST be possible to change permissions to individually identifiable information at any time, and the updated permissions MUST be honoured immediately. [TAS3, 6]

- Access Privileges

8. The specific permissions supported by the systems access control policy MUST be: read, write (add, upload, create, update, delete, rename), execute, export and download.
9. Every role in the system MUST have a set of the permissions existing in the access control policy.
10. The administrator user MUST have all the access rights.
11. Only the data stewards SHOULD upload the study data.
12. The non-administrator users SHOULD have access only to data elements and SHOULD not have access to the administrative elements (e.g. users, user roles, access rules).
13. The system MUST grant access to the systems data only for users registered in the system.
14. The research institutions MUST be able to write data into their DMZ partitions.
15. The research institutions MUST be forbidden to read data from the DMZ.
16. The Quality Checker (person or tool) MUST be able to read data from the DMZ.
17. The Quality Checker MUST be forbidden to write data in the DMZ.
18. The Quality Checker MUST reject the incorrect anonymised data.
19. The Quality Checker MUST communicate the incorrect anonymisation of the data to the research institute owner of this data.
20. In case of incorrect anonymised data, the research institute MUST re-anonymise its data and re-upload it in its DMZ partition.
21. In case of incorrect anonymised data followed by no re-anonymisation and no re-upload of data, the research institute's anonymised data MUST not arrive in the "globally accessible shared translational database".
22. Not all the members from an organisation SHOULD necessarily have the same access privileges to the "globally accessible shared translation research database".
23. Access to the "globally accessible shared translation research database" MUST be limited to the people who have statistical analysis knowledge.
24. The DMZ MUST be separated into landing zones, which are mapped to the organisations that are uploading anonymised data.
25. The access control policy of the DMZ MUST forbid the organisations to have access to any other part of the DMZ, except their DMZ landing zone.

- User Roles / Document Classes / Permissions Changes

26. The "globally accessible shared translational research database" administrator MUST be able to create and administrate the user roles inside his/her study.
27. The system MUST support the changing of user roles / document classes / permissions by the administrator.
28. The system MUST support the mapping of its users to different user roles inside and outside of a study.

29. The system MUST limit the collection of data to what is needed. [TDL, 17]

- Access Control Granularity

30. The granularity of the access control MUST be at study level.

C.2.3 Confidentiality

- Methods of providing data confidentiality

1. The confidentiality of the data MUST be protected using access control and anonymisation.
2. The anonymisation of the data MUST be done by the research institute that owns the data.
3. The transfer of data from the research institutes to the "globally accessible shared translational research database" MUST be secured.
4. In case of losing a study's data anonymity, the study MUST become invalid and the study MUST be anonymised again.
5. At a user's login in the system, the user's password SHOULD be encrypted over the network.

- General

6. Disclosure of any data (even anonymous data) MUST be avoided.
7. The most sensitive data MUST be the personal identifiable information and the Intellectual Property (IP) data.
8. Adding more data to a study SHOULD not decrease the level of anonymity of the dataset.

C.2.4 Trust Requirements

1. The Trust-PDP MUST support user-selected policies. [TAS3, 8]
2. The Trust-PDP MAY support trust decisions based on dynamic data such as Key Performance Indicators. [TAS3, 9]
3. The Trust-PDP SHOULD be able to request required credentials from external sources. [TAS3, 12]
4. The authorisation component MAY be able to make use of information from the Trust component when making an authorisation decision. [TAS3, 39]
5. The architecture MUST provide a facility (PEP) to determine when trust policies need to be evaluated and call the appropriate component (the Trust PDP). [TAS3, 43]
6. The Trust-PDP SHOULD provide sufficient performance. [TAS3, 45]
7. Audit Guard Service: a service SHOULD be able to log outgoing requests, policies, and transactions. [TAS3, 49]

8. All components of the system MUST be constructed and configured in such a way that no single failure of any component can reduce the security and trustworthiness level of any other component or the system as a whole. NOTE this does not mean that the system MUST be reliable in terms of uptime or throughput performance. [TAS3, 50]
9. If any component encounters trouble, the system MUST be able to correctly identify the root cause of the problem using log escalation and other techniques. [TAS3, 51]
10. To improve trustworthiness, the system MUST have a clear time limit on failure fix attempts. Beyond the time limit, a routine rollback to a known-secure state MUST be executed. [TAS3, 52]
11. Wherever possible, the system MUST deploy standard software components to perform well-defined tasks instead of integrating these tasks into custom software. NOTE the reasoning is that this SHOULD reduce the chance of error and lowers the maintenance burden on the developers. [TAS3, 53]
12. The relevant interface specifications SHOULD be published on a revision-controlled system with mandatory subscription, so that any changes can be guaranteed to reach the relevant developers. [TAS3, 54]
13. All components of the system SHOULD be built using strict quality assurance procedures, with evidence of the process available. [TAS3, 55]
14. To improve trustworthiness, the system SHOULD implement Audit Guards in all relevant places as early as possible. [TAS3, 57]
15. To improve trustworthiness, the system SHOULD implement constellation-wide testing facilities as early as possible. These facilities will be used to do early certification testing and run-time monitoring. [TAS3, 58]
16. To improve trustworthiness, the system MUST have clearly distinct roles and duties for developers and administrators. [TAS3, 59]
17. The system MAY have means to indicate trustworthiness of data, products and services. [TDL ,5]
18. The system MUST ensure enforcement, auditing and rollback in case of failure. [TDL, 14]

C.3 Non-Functional Requirements

1. All the systems studies MUST comply with the Good Clinical Practice (GCP) requirements.
2. The DMZ MUST respect all data protection regulations and laws from all five Movember regions (Australasia, Europe, UK, Canada, USA).
3. The changes made to user roles / document classes / permissions MUST take effect within five minutes.

4. The systems access control policy SHOULD provide flexibility and dynamicity in order for new user roles to be created anytime within five minutes.

C.4 Out of Scope Items

1. The organisations MUST have the patients consent in order to use their data in translational research.

NOTE
It is assumed that obtaining and recording consent is outside the scope of the system. If an investigator can obtain data from the system, then this implies that consent must have been granted.

C.5 Assumptions

- Nil.

Appendix D

Requirements: Collaborative PACS (Picture Archiving and Communication System)

Glossary

Acronym	Full name/Description
ABC4Trust	An EU-funded research and development project advancing the federation and interchangeability of technologies supporting trustworthy and at the same time privacy-preserving Attribute-based Credentials.
PEP	A PEP passes a complete set of validated and aggregated attributes to a Policy Decision Point for the latter to make access control decisions
TAS3	Trusted Architecture for Securely Shared Services. A project aiming to have a European-wide impact on services based upon personal information, which is typically generated over a human lifetime and therefore is collected and stored at distributed locations and used in a multitude of business processes.
TDL	Trust in Digital Life. An FP7 funded research community for the research and development of trustworthy ICT solutions.
Trust-PDP	Trust Policy Decision Point. Responsible for coordinating and combining trust services for nested policies.

D.1 Description

D.1.1 Introduction and background

PACS (picture archiving and communication system) is a medical imaging technology that provides storage of, and convenient access to, images from multiple modalities. PACS has four main uses, namely: (i) hard copy replacement; (ii) remote access that provides capabilities of off-site viewing and reporting enabling practitioners in different physical locations to access the same information simultaneously; (iii) an electronic image integration platform that provides interfacing with other medical automation systems such as Electronic Medical Record (EMR) systems; and (iv) radiology workflow management.

Given that PACS must store and transmit sensitive data, accessed from different organisations (hospitals) and by different users in a distributed system, it is of paramount importance to ensure proper user authentication and authorisation. In order to comply with legislation, patient permissions to access data must be properly regulated (patient digital consent management).

Access to medical information such as PACS records is usually regulated by access control policies, that specify which parties and under what conditions those parties may access information.

D.2 Functional Requirements

Some of the PACS requirements are derived or taken directly from the TAS3, ABC4Trust and TDL security requirements. These requirements are signalled by references to the relevant sources (from the AU2EU State-of-the-Art Requirements document).

D.2.1 Authentication

- Authentication method

  1. Users SHOULD be able to login using the credentials from their hospital.
  2. Users MUST be able to authenticate to services using their existing authentication credentials without needing to be issued with new authentication credentials. [TAS3, 27]
  3. Users MUST be enrolled with an Identity Provider in order to use the services of its mutually trusted Service Providers. [TAS3, 28]
  4. Users SHOULD be allowed to enrol with multiple Identity Providers under different unrelated identities. [TAS3, 29]
  5. In case of strong authentication, a hardware card which generates presentation tokens based on privacy attribute based credentials solution MAY be used. [ABC4Trust,1]
  6. Privacy attribute based credentials MUST be able to be bound to the hardware card and/or to the user. [ABC4Trust,3]
  7. The user MUST not be able to manipulate the presentation tokens or the privacy attribute based credentials without damaging their integrity. [ABC4Trust, 4]
  8. Privacy attribute based credentials MUST be stored on the hardware card. [ABC4Trust, 5]

9. The user MUST be able to read all contents of his hardware token. [ABC4Trust, 7]
10. If the hardware token contains a users secret, the user MUST not be able to read the users secret. [ABC4Trust, 7]
11. The system MUST prevent the users from generating tokens from attributes not certified by their own privacy attribute based credentials. [ABC4Trust, 17]
12. A replay of the same presentation token MUST not be allowed by the system. [ABC4Trust, 19]
13. The audit logs MUST never reveal the values of non-public keys and secrets. [ABC4Trust, 21]

- User accounts

14. People SHOULD be able to use their existing accounts when taking part in processes of organisations they did not have contact with before. [TAS3, 1]
15. The system SHOULD minimise the amount of data collected from the users. [TDL, 17]

D.2.2 Authorisation

- Access Control Policy

1. The system's access control policy MUST support role-based access control (individuals are assigned to roles and roles are assigned different permissions).
2. The system's access control policy MAY support the concept of groups of users.
3. A patien's consent MUST be collected and enforced before any patient data enters the PACS database.
4. Patient consent policy MUST be enforced every time patient data are accessed.
5. The geographical localisation of the user who wants to get access to data MUST be used in authorisation enforcement.
6. The policies MUST be stored in or next to the PACS database.
7. The system MUST support the fact that not all the users belonging to the same hospital have the same access privileges to a patient's data.
8. It MUST be possible to specify separation of duty constraints, i.e. that some tasks MAY not be performed by the same person. [TAS3, 5]
9. It MUST be possible to change permissions to individually identifiable information at any time, and the updated permissions MUST be honoured immediately. [TAS3, 6]
10. Service Providers MUST be able to set the policies (authentication, authorisation and trust) that control access to the resources under their control. [TAS3, 30]
11. Users SHOULD be able to set the policies that control access to their data, and this policy SHOULD stay with their data regardless of where this data is subsequently held. [TAS3, 31]

12. Administrators MUST be able to set the policies that control legal access to data. [TAS3, 32]
13. The authorisation system MUST support the setting of multiple authorisation policies by multiple authorities and be able to resolve any conflicts between these in a mutually acceptable manner. [TAS3, 33]
14. The authorisation system MAY be able to support history based access control decision making i.e. where the current decision is based on both the current context and previous decisions. [TAS3, 34]
15. The system MUST support a tamper proof audit capability in which the audit logs are visible to trusted third parties. [TAS3, 35]
16. The Trust-PDP MUST support user selected policies. [TAS3, 8]
17. The Trust-PDP MAY support trust decisions based on dynamic data such as Key Performance Indicators. [TAS3, 9]
18. The Trust-PDP SHOULD be able to request required credentials from external sources. [TAS3, 12]
19. The Trust-PDP SHOULD be open and flexible enough to support multiple systems. [TAS3, 46]
20. The system MUST provide tracking of location with differentiated access to the data. [TDL, 11]

- User Roles / Document Classes / Permissions Changes

21. The system MUST support the changing of user roles / document classes / permissions by the administrator.
22. The systems access control policy SHOULD provide flexibility and dynamicity in order for new user roles to be created anytime.
23. The system MUST support the mapping of its users to different user roles inside and outside of a study.
24. The system MUST limit the collection of data to what is needed. [TDL, 17]

- Access Control Granularity

25. The system MUST support different access control policies for data with different sensitivity levels.
26. The access control policy SHOULD support various levels of granularity aligned with data granularity (e.g. up to very fine grained such as database cells).

D.2.3 Confidentiality

1. The confidentiality of the data MUST be protected using policy enforcement and/or encryption.

2. If the data anonymity of a study is compromised, the study MUST become invalid and the study MUST be anonymised again.

3. Disclosure of any data (even anonymous data) MUST be avoided.

D.2.4 Auditability

1. The information that is sent to the audit function SHOULD be dependent upon the state of the system, and events (external or internal) MUST be able to dynamically effect what information is sent to the audit function.[TAS3, 42]
2. The system MUST support a tamper-proof audit capability in which the audit logs are visible to trusted third parties. [TAS3, 36]
3. All system components MUST be able to write to the audit function. [TAS3, 38]

D.2.5 Trust Requirements

1. The authorisation component MAY be able to make use of information from the Trust component when making an authorisation decision. [TAS3, 39]
2. The architecture MUST provide a facility (PEP) to determine when trust policies need to be evaluated and call the appropriate component (the Trust PDP). [TAS3, 43]
3. The Trust-PDP SHOULD provide sufficient performance. [TAS3, 45]
4. Audit Guard Service: a service SHOULD be able to log outgoing requests, policies, and transactions. [TAS3, 49]
5. All components of the system MUST be constructed and configured in such a way that no single failure of any component can reduce the security and trustworthiness level of any other component or the system as a whole. NOTE this does not mean that the system MUST be reliable in terms of uptime or throughput performance. [TAS3, 50]
6. If any component encounters trouble, the system MUST be able to correctly identify the root cause of the problem using log escalation and other techniques. [TAS3, 51]
7. To improve trustworthiness, the system MUST have a clear time limit on failure fix attempts. Beyond the time limit, a routine rollback to a known-secure state MUST be executed. [TAS3, 52]
8. To improve trustworthiness, the system SHOULD implement Audit Guards in all relevant places as early as possible. [TAS3, 57]
9. To improve trustworthiness, the system SHOULD implement constellation-wide testing facilities as early as possible. These facilities will be used to do early certification testing and run-time monitoring. [TAS3, 58]
10. To improve trustworthiness, the system MUST have clearly distinct roles and duties for developers and administrators. [TAS3, 59]
11. The system MAY have means to indicate trustworthiness of data, products and services. [TDL,5]
12. The system MUST ensure enforcement, auditing and rollback in case of failure. [TDL, 14].
13. The PACS users SHOULD have the ability to choose between different levels of trust and privacy. [TDL, 16]

D.2.6 Miscellaneous

1. The system MUST provide users with a) an understanding what is going on, and b) the ability to deal with privacy issues themselves. [TDL, 2]

D.3 Non-Functional Requirements

1. The collaborative scenario SHOULD not affect more than 10% of the PACS systems performance.
2. Modifications of the policies MUST be reflected immediately (5 seconds) in all policy enforcement actions from the whole Collaborative PACS system.
3. Database segmentation MAY be a solution for limiting exposure in case of systems breach.
4. The architecture SHOULD provide secure data repositories. [TAS3, 24]
5. Processes MUST be able to react to and recover from security violations in a secure way. [TAS3, 26]
6. Wherever possible, the system MUST deploy standard software components to perform well-defined tasks instead of integrating these tasks into custom software. NOTE the reasoning is that this SHOULD reduce the chance of error and lowers the maintenance burden on the developers. [TAS3, 53]
7. The relevant interface specifications SHOULD be published on a revision-controlled system with mandatory subscription, so that any changes can be guaranteed to reach the relevant developers. [TAS3, 54]
8. All components of the system SHOULD be built using strict quality assurance procedures, with evidence of the process available. [TAS3, 55]

D.4 Out of Scope Items

- Nil.

D.5 Assumptions

- Nil.