HAL open science

Data Governance and Transparency for Collaborative Systems

Rauf Mahmudlu, Jerry Den Hartog, Nicola Zannone

To cite this version:

Rauf Mahmudlu, Jerry Den Hartog, Nicola Zannone. Data Governance and Transparency for Collaborative Systems. 30th IFIP Annual Conference on Data and Applications Security and Privacy (DBSec), Jul 2016, Trento, Italy. pp.199-216, 10.1007/978-3-319-41483-6_15 . hal-01633673

HAL Id: hal-01633673

https://inria.hal.science/hal-01633673

Submitted on 13 Nov 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Data Governance & Transparency for Collaborative Systems

Rauf Mahmudlu, Jerry den Hartog, and Nicola Zannone

Eindhoven University of Technology
r.m.o.mahmudlu@student.tue.nl, {j.d.hartog,n.zannone}@tue.nl

Abstract. As social networks, shared editing platforms and other collaborative systems are becoming increasingly popular, the demands for proper protection of the data created and used within these systems grows. Yet, existing access control mechanisms are not suited for the challenges imposed by collaborative systems. Two main challenges should be addressed: collaborative specification of permissions, while ensuring an appropriate levels of control to the different parties involved, and enabling transparency in decision making in cases where the access requirements of these different parties are in conflict. In this paper we propose a data governance model for collaborative systems, which allows the integration of access requirements specified by different users based on their relation with a data object. We also study the practical feasibility of enabling transparency by comparing different deployment options for transparency in XACML.

1 Introduction

Collaborative systems such as social networking websites, document sharing/editing platforms and audio/video conferencing tools, are gaining increasing popularity over the years. These systems provide an environment wherein users can collaborate and share information. This information, however, can be sensitive and, thus, needs to be protected from unauthorized access and accidental loss or modification.

Access control is widely used to protect sensitive information. Access control mechanisms rely on policies defining which actions users are allowed to perform on data objects. However, existing authorization mechanisms are not able to deal with the security demands of collaborative environments [21]. In particular, we identify two main drawbacks that limit their application to collaborative systems: the lack of *(i)* a data governance model for shared data objects and *(ii)* transparency in decision making.

Most access control mechanisms assume that data objects are under the control of a single entity (e.g., the system or the owner). However, in collaborative systems several users can contribute to the creation, governance and management of data [3, 7]. For instance, data can be provided by one or more users, can be stored by some other user, and refer to yet other users where each of these users retains some level of authority on the data. In particular, each user can define its own authorization requirements for the protection of data. Therefore, we need a way to combine those requirements in order to define the policy ultimately regulating the access to data.

Several approaches for policy combination [8, 11, 12, 14] and integration [13] have been proposed. These approaches provide strategies to combine policies specified by

different entities and automatically resolve policy conflicts at evaluation time based on predefined priorities between decisions or based on the policy structure. However, they consider every user 'equally' and they do not account for the relation of users with the data to be protected in order to determine how user policies should be combined.

Although the use of these strategies is necessary to guarantee the proper functioning of the system as a conclusive decision has to be made (either allowing or denying the access to the data), it results in a decision making process that is non-transparent to users. Every user expects its policies to be enforced by the system. This, however, is often not possible, for instance when users specify conflicting authorization requirements for the same resource. To resolve policy conflicts, policy combination strategies sacrifice some policies to reach a conclusive decision. Authorization mechanisms make decisions in a blackbox manner [5] and, thus, users are often unaware whether their policies have actually been enforced. This lack of confidence may reduce the level of trust users have towards the system and thus users' willingness to engage in collaboration.

In a previous work [2] we have introduced the notions of *archetype* and *policy mismatch* to address these issues. Archetypes are used to represent the relation of users with a given data object. Policy mismatches are used to identify the difference between the authorization requirements of single users and the final decision enforced by authorization mechanisms. We have also shown how the notion of policy mismatch can be used as a baseline for the realization of transparent authorization mechanisms which increases user awareness about access decision making.

This paper extends our previous work in two directions. First, we propose a data governance model for collaborative systems, which allows the integration of authorization requirements specified by different users based on their relation with a data object. In particular, the governance model provides a general framework to reason on the level of authority that users have over shared data and allows the use of existing policy combination and integration strategies to resolve policy conflicts. Moreover, we investigate the feasibility of transparency in existing authorization mechanisms. In particular, we have developed a transparency service that has been deployed in SAFAX [10], an XACML-based framework offering authorization as a service. A main feature of SAFAX is that all the components of the XACML reference architecture are designed as loosely coupled services. We exploit the flexibility provided by this design to evaluate the impact of the transparency service with respect to different deployment models.

The remainder of the paper is organized as follows. The next section introduces preliminaries on XACML. Section 3 presents our approach to shared data control. Section 4 discusses the problem of decision mismatches, and Section 5 describes the design, implementation and deployment of the transparent service. Section 6 presents experimental results. Finally, Section 7 discusses related work, and Section 8 concludes the paper and provides directions for future work.

2 Preliminaries

This section provides preliminaries on XACML [14], the *de facto* standard for the specification and enforcement of access control policies. This work is based on XACML v2. However, it can be easily adapted to comply with XACML v3.

2.1 Policy Language

XACML provides an attribute-based language that allows the specification of composite policies by using three policy elements: *policy sets*, *policies* and *rules*. Policy sets comprise a list of policy sets and policies; policies comprise a list of rules. Rules specify an *effect*, i.e. whether an access request should be allowed (Permit) or denied (Deny). Each policy element has a (possibly empty) *target* which restricts the applicability of the policy element. The target is specified in terms of attributes characterizing the subject, resource, action and environment and denotes the access requests covered by the policy element. A rule may additionally have a *condition*, i.e. a predicate that must be satisfied for a rule to be applicable. Policy sets and policies can also be associated with *obligations*, i.e. mandatory requirements that have to be fulfilled.

The evaluation of an access request against a policy element results in an access decision. If the request matches both the target and condition of a rule, the rule is applicable to the request and yields the decision specified by its effect, either Permit or Deny. If the rule is not applicable, a NotApplicable decision is returned. If an error occurs during evaluation, an Indeterminate decision is returned. Each composite policy element (i.e., a policy set or a policy) specifies a *combining algorithm* that is used to combine the decisions of its comprising elements. XACML provides a number of combining algorithms: permit-overrides (pov), deny-overrides (dov), first-applicable (fa) and only-one-applicable (ooa). These algorithms evaluate composite policies based on the order of the policy elements and priorities between decisions. Hereafter, we use the following abstract notation to represent the policy evaluation process in XACML: $\mathcal{P}$ denotes the set of XACML policies, $\mathcal{Q}$ the set of access requests, and function $[\![p]\!] : \mathcal{Q} \rightarrow \{\mathsf{Permit}, \mathsf{Deny}, \mathsf{NotApplicable}, \mathsf{Indeterminate}\}$ denotes policy evaluation, i.e. $[\![p]\!](q)$ is the decision according to a policy $p \in \mathcal{P}$ for a request $q \in \mathcal{Q}$.

2.2 XACML Architecture

The XACML reference architecture is shown in Figure 1. Access requests are intercepted by the *Policy Enforcement Point* (PEP). Upon receiving an access request, the PEP forwards the request to the *Context Handler* (CH) which, after translating the request from the application's native format to XACML, sends it to the *Policy Decision Point* (PDP) for evaluation. The PDP fetches the policies from the *Policy Administration Point* (PAP). If additional attributes are required to evaluate the request, the PDP queries the CH for such attributes. The CH retrieves these attributes from the *Policy Information Point* (PIP) and sends them to the PDP. The PDP evaluates the request against the policies and returns a response specifying the access decision (and possibly a set of obligations to be fulfilled) to the CH. The CH sends the response to the PEP, which is responsible for the enforcement of the decision and the fulfillment of obligations.

3 Shared Data Control

In collaborative systems like social media and document sharing platforms, data objects can be under the control of multiple stakeholders. The level of authority that each stakeholder has on a shared data object depends on its relation with the object. In this section

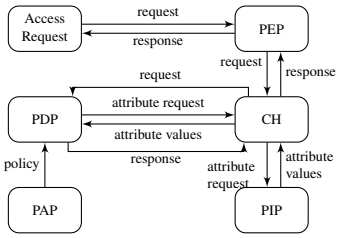


Fig. 1: XACML Architecture

we discuss the problem of shared data control and propose an approach to regulate the access to data by taking into account both the authorization requirements of the stakeholders related to the data and their relationship with the shared data. We start by introducing a scenario in healthcare that is used as a running example throughout the paper.

Example 1. A University Medical Center (UMC) provides medical treatment for a variety of diseases. The UMC also has an advanced research program, and several researchers conduct clinical research studies within the UMC. Patient data are stored in a central database at the UMC. The UMC is responsible for guaranteeing the security of patient data and for determining the purposes and means of its processing. Different departments at the UMC can define policies to regulate the access to patient data. Here, we consider two such departments: the Security Department and the Data Center. The Data Center manages the UMC database and is mainly concerned that medical, research and administrative staff of the UMC have access to the data they need to perform their duties. On the other hand, the Security Department mainly focuses on the protection of patient data and on the compliance with regulations and laws that are in place. Our scenario focuses on Alice and Caroline, two monozygotic twins, who both rely on UMC's services for treatment. Caroline has also engaged in a clinical trial and shared her genetic information with the UMC for research purposes.

Privacy is a highly regulated subject, especially in healthcare. Most countries have regulations and laws in force, which impose stringent requirements on the collection and processing of personal data [6]. To explicitly model the access requirements defined by privacy regulations, we introduce a *Regulatory Body* as a stakeholder in our scenario.[1] This entity issues and revises regulations to protect the privacy of citizens. Privacy regulations like the EU Directive on data protection (Directive 1995/46/EC) require the creation of an independent authority to protect the fundamental rights of citizens. This authority, hereafter referred to as National Privacy Authority, has the task of overseeing the compliance of organizations with privacy regulations. Moreover, it can prohibit unlawful or unfair data processing operations. Next to the National Privacy Authority, we also consider an Ethical Medical Committee of the Ministry of Health. This entity defines requirements on the use of medical data, especially for research purposes.

Each aforementioned stakeholder can specify requirements to regulate the access and usage of data. These requirements are summarized in Table 1.

[1] Note that legal requirements can also define the relation between stakeholders. In the next section we will discuss how these requirements can be accommodated in the framework.

Stakeholder	Access Requirements
Alice	Her treating doctors and nurses can access her medical information.
	Any other access to her information is denied.
Caroline	Doctors and nurses can access her medical information.
	Researchers can read her genetic information.
	Any other access to her information is denied.
UMC Data Center	Doctors and nurses can access patient information.
	Researchers can access patient information.
UMC Security Department	Medical staff can access patient data to provide medical treatment.
	Technicians can access and modify patient data for maintenance purposes.
Regulatory Body	Data subjects can access their medical information.
	Personal data shall be collected and processed only if the data subject has given his explicit consent to their processing.
	Access is allowed without data subject's consent to comply with a legal obligation imposed upon the controller.
	Access is allowed without data subject's consent to protect the data subject's vital interests.
National Privacy Authority	Unlawful and unfair data processing operations are forbidden.
Ethical Medical Committee	Researchers can only access anonymized patient information.

Table 1: Access Requirements

3.1 Data Governance Model

In collaborative systems, multiple stakeholders can contribute to the creation and management of data objects. Each stakeholder related to a data object should retain some authority on the object. However, not all these stakeholders might have the same level of authority. The degree of authority a user has depends on its role with respect to the data. Thus, the actual permissions on shared data should be defined by taking into account both stakeholders' access requirements and their relation with the data. In this section, we investigate a general framework to explicitly express the relations between stakeholders and data objects as well as to prioritize such relations.

To characterize the relation between stakeholders and shared data, we use the notion of archetype proposed in [2]. The archetypes for a shared data object capture the roles that stakeholders can have with respect to the object. The role determines the extent of control over the object. In this work we introduce the notion of archetype hierarchy to reflect the level of authority that users have on shared resources.

Definition 1. *Let A be the set of archetypes for a shared data object o. An* archetype hierarchy *H has the form:*

$$\begin{aligned} H &= L \mid (L, pr, H) \\ L &= (\sigma, [a_1, \ldots, a_n]) \\ pr &= t \mid + \mid - \end{aligned}$$

A level L consists of a set of archetypes $a_1, \ldots, a_n \in A$, whose requirements are combined using an intra-level aggregator σ. An archetype hierarchy H is (recursively) built over levels by concatenating a level with a hierarchy according to a given priority pr that can be total (denoted by t), positive (denoted by $+$) or negative (denoted by $-$).

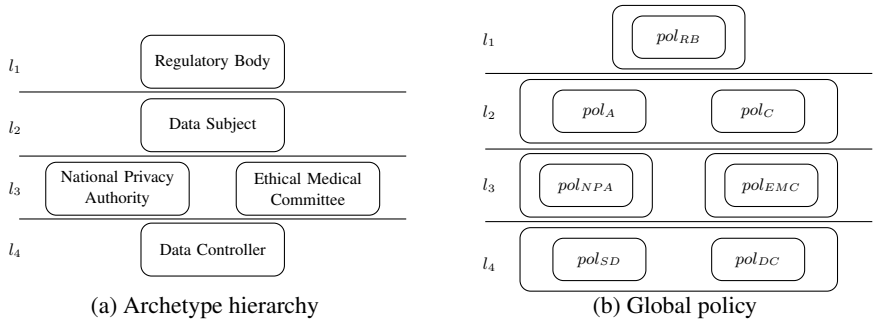
Intuitively, a level groups those archetypes that have the same level of authority on shared data. An intra-level aggregator specifies how the requirements of the stakeholders associated to the archetypes in a level should be evaluated. Our framework does not pose restrictions on the intra-level aggregator that can be used. In the next section we provide some examples of intra-level aggregators and discuss how they can be realized.

Example 2. Consider the genetic information provided by Caroline in the scenario of Example 1. We identify two main archetypes for this information: *Data Controller* and *Data Subject*. The Data Controller is the entity responsible for the security of the data and defines who can access a data element and how data can be processed. In our scenario, the UMC plays the role of Data Controller for Caroline's genetic information. In particular, the UMC Data Center and Security Department are two instances of the Data Controller. The Data Subject is the person to whom the information refers. In the scenario, Caroline is the Data Subject for her genetic information. In addition, given the twin relationship between Alice and Caroline, we also consider Alice as the Data Subject for the genetic information provided by Caroline. Next to these archetypes, we define an archetype for each of the other stakeholders in the scenario, namely Regulatory Body, National Privacy Authority and Ethical Medical Committee.

In an archetype hierarchy, levels are ordered according to the degree of authority that the archetypes forming a level have. We distinguish three types of priorities between levels: total, positive and negative. Total priority indicates that the access requirements associated to the higher level always override the ones associated to lower levels. However, in some cases only the positive access requirements (i.e., access requirements defining positive authorizations) associated to the higher level should take precedence; otherwise, the access requirements defined by stakeholders at the lower level should also be evaluated. This is achieved using the positive priority. Negative priority is the dual of positive priority where only negative requirements from the higher level take precedence.

Example 3. The archetypes for our running example, identified in Example 2, can be organized in a hierarchy. Fig. 2a presents a graphical representation of this hierarchy. Regulatory Body has the highest priority. The next level comprises the Data Subject, followed by a level formed by the National Privacy Authority and the Ethical Medical Committee. The lowest level is formed by the Data Controller. In order to comply with data protection regulations and to satisfy the intrinsic characteristics of the roles, the following priorities are defined between levels:

- The Regulatory Body has the right to override the decisions of the Data Subjects to permit access to patients' medical records, e.g. to protect their vital interests or comply with legal obligations [6]. Therefore, a positive priority is used between the first and second level.
- Data Subjects have the right to determine who can (or cannot) access and process their information. However, even if they permit access to their information, the National Privacy Authority and the Ethical Medical Committee hold the right to deny it if the request is not in compliance with their requirements. Such a requirement is achieved through a negative priority between the Data Subject and the lower level.



(a) Archetype hierarchy (b) Global policy

Fig. 2: Data Governance Model and Instantiation for the scenario in Example 1

– The National Privacy Authority and Ethical Medical Committee can influence how the Data Controller processes personal data. In particular, they can deny an unlawful or unfair access, or permit the access for research purposes regardless of the Data Controller's requirements. A total priority between the levels can be used to achieve this requirement.

3.2 Data Governance Instantiation

Access control policy languages like XACML allow stakeholders to express their access requirements as policies and provide means to combine these policies in a single policy (hereafter referred to as the *global policy*), which is used to determine the actual permissions on shared data. In this section, we present how the global policy can be created from user policies taking into account the archetype hierarchy. We first introduce a grammar for the specification of the global policy. This grammar is inspired by XACML, thus making the encoding into XACML policies straightforward.

Definition 2. *The global policy P_H is constructed upon the following grammar:*

$$\begin{aligned} P_H &= P_L \mid (\mathsf{fa}, [P_L, P_H]) \mid (\mathsf{pov}, [P_L, P_H]) \mid (\mathsf{dov}, [P_L, P_H]) \\ P_L &= (ca, [P_a, \ldots, P_a]) \\ P_a &= (ca, [p_1, \ldots, p_n]) \end{aligned}$$

where $p_1, \ldots, p_n$ are user policies and ca represents a policy combining algorithm.

An archetype policy P_a combines the policies of those users who are associated to an archetype a. To this end, every archetype is associated with a policy combining algorithm that determines how the policies defined by the stakeholders having such an archetype are combined. A level policy P_L combines the policies associated to the archetypes in a level L. The combining algorithms used to construct archetype and level policies should reflect the security and privacy needs for the specific domain. In particular, the combining algorithm for level policies should reflect the constraints on the combination of archetypes in a level as given in the archetype hierarchy (Definition 1). Note that our policy language does not impose any restriction on the policy combining

algorithms to be used to combine user policies and archetype policies.[2] For instance, archetype/level policies can make use of the standard XACML combining algorithms (see Section 2) or more advanced combining algorithms such as the consensus and majority combining algorithms defined in [11]. The global policy is recursively built over level policies. This is necessary to account for the use of different priority between levels in the archetype hierarchy. Priorities are encoded in terms of combining algorithms. In particular, the total, positive and negative priorities are encoded using first-applicable (fa), permit-overrides (pov) and deny-overrides (dov), respectively.

Next we define how the global policy is constructed from the archetype hierarchy and user policies.

Definition 3. *Let A be the set of archetypes for an object o and U the set of users. Let $UA \subseteq U \times A$ be the user-archetype assignment, i.e. $(u, a) \in UA$ iff user u has archetype a. Let $\mathcal{P}$ be the set of user policies and let p_u denote the policy of user u. We denote by $\mathsf{A2ca}(a)$ the combining algorithm ca specified for archetype a. To combine user policies according to the archetype hierarchy, we first create* archetype policies*:*

$$\mathsf{A2P}(a) = (\mathsf{A2ca}(a), [p_{u_1}, \ldots, p_{u_m}])$$

where $u_1, \ldots, u_m$ are the users such that $(u_1, a), \ldots, (u_m, a)$ are in UA. Next, archetype policies are combined to form level policies*:*

$$\mathsf{L2P}((\sigma, [a_1, \ldots, a_n])) = (ca, [\mathsf{A2P}(a_1), \ldots, \mathsf{A2P}(a_n)])$$

where ca is the combining algorithm realizing the intra-level aggregator σ. The global policy *is obtained by recursively combining level policies with respect to the priorities between levels:*

$$\begin{cases} \mathsf{H2P}(L) = \mathsf{L2P}(L) \\ \mathsf{H2P}((L, pr, H)) = (\mathsf{pr2CA}(pr), [\mathsf{L2P}(L), \mathsf{H2P}(H)]) \end{cases}$$

where

$$\mathsf{pr2CA}(t) = \mathsf{fa} \qquad \mathsf{pr2CA}(+) = \mathsf{pov} \qquad \mathsf{pr2CA}(-) = \mathsf{dov}$$

In the next example we illustrate how to derive a global policy from the archetype hierarchy and user policies based on our running example.

Example 4. Fig. 2b shows the structure of global policy G obtained by instantiating the archetype hierarchy in Fig. 2a based on the scenario given in Example 1. Formally, the global policy can be represented as follows:

$$G = \mathsf{pov}(pol_{RB}, \mathsf{dov}(\mathsf{pov}(pol_A, pol_C), \mathsf{fa}(\mathsf{wc}(pol_{NPA}, pol_{EMC}), \mathsf{dov}(pol_{SD}, pol_{DC}))))$$

Here we assume that the policies specified by the data subjects (i.e., Alice and Caroline) are combined using permit-overrides, i.e. access to the data is granted if at least

[2] Although any combining algorithm can be used to combine user policies and archetype policies, the use of noncommutative algorithms can have undesired effects. In fact, these algorithms often represent a priority between policies based on their order (e.g., first-applicable in XACML), whereas there is no order within an archetype or a level.

one of the data subjects permits the access. The policies of the UMC Security Department and Data Center are combined using deny-overrides. Finally, we combine the policies of the National Privacy Authority and the Ethical Medical Committee using the weak-consensus algorithm as defined in [11]. According to this algorithm, user policies should not conflict with each other: Permit a request if some user policies permit a request, and no user policy denies it; Deny a request if some user policies deny a request, and no user policy permits it; otherwise Indeterminate should be yielded.

4 Policy Mismatches

In the previous section, we have shown how the policies of different stakeholders can be combined by taking into account their relationship with the resource to be protected. Ideally, the authorization system should enforce the access requirements of all stakeholders. However, this is not always possible. In fact, users can have conflicting authorization requirements, which results in conflicting policies.

Many access control mechanisms like XACML use policy combining algorithms to automatically resolve policy conflicts. Although solving conflicts is necessary for an authorization mechanism to make a conclusive decision, it makes the decision making process non-transparent to users. Users expect their policies to be enforced by the authorization system; however, in practice, their policies can be overridden by the policies of other entities. The main problem is that, in most existing authorization systems, policy conflict resolution is embedded in the policy evaluation process and, thus, policy conflicts are not identified and/or recorded. This makes users unaware whether their policies have actually been enforced.

We argue that the lack of transparency can affect the collaboration among users and, in particular, their willingness of sharing sensitive information needed for the success of the collaboration. Below we exemplify this issue using our running example.

Example 5. As shown in Example 1, each stakeholder has certain authorization requirements over the genetic information provided by Caroline. Suppose that David, a researcher at the UMC, requests access to this information. Based on the global policy in Example 4 and access requirements in Table 1, the authorization system allows David to access the information. If we look at the requirements of the single users, we have that: the enforced decision is consistent with Caroline's and the UMC Data Center's policy; however, access should have been denied according to Alice's policies; finally, the Regulatory Body's policy returns a NotApplicable as it delegates the Data Subject the authority to decide whether its data can be used for research purposes and thus does not define a specific policy about researcher accessing genetic information. We can observe that Alice's access requirements are not enforced. This can reduce her trust towards the UMC and, thus, can make her reluctant to share information in the future.

We use the notion of *policy mismatch* introduced in [2] to capture policy conflicts.

Definition 4. *Let $p_1, \ldots, p_n$ be the policies of n users and p the global policy obtained by combining such policies. Given an access request q, a user i (with $i \in \{1, \ldots, n\}$) has a mismatch if $[\![p]\!](q) \neq [\![p_i]\!](q)$.*

A mismatch occurs when the decision enforced by the authorization system differs from the decision obtained evaluating the policy of a user. Likely, only mismatches where a user's policy is applicable (i.e., $[\![p_i]\!] \neq \mathsf{NotApplicable}$) are relevant for the user. However, we do not restrict the (type of) mismatches that can be reported to users. In particular, we allows each user to specify mismatches preferences, indicating the types of mismatches the user wants to be notified (see Section 5.1). In the next section, we show how the notion of mismatch can be used to augment the XACML reference architecture with a transparency service while being compliant with the XACML specification.

5 Transparency Service

The goal of this work is to enable collaborations between stakeholders in a trusted, secure and privacy-preserving way. Sharing resources and managing access to them are essential for such collaborations. However, as shown in the previous section, stakeholders may have conflicting authorization requirements. This section presents a *transparency service*, which aims to make the stakeholders engaged in a collaboration aware of these conflicts and how they are resolved by the authorization system.

The transparency service has been designed to be fully compliant with the XACML standard. This ensures that the service can be used within existing XACML implementations without these implementations being modified. In the remainder of the section, we discuss the design and implementation of the transparency service as well as possible deployment configurations within the XACML reference architecture.

5.1 Service Design

The transparency service aims to detect mismatches between the decision enforced by the authorization system and the access requirements of a certain stakeholder. Any mismatch found is then reported to the stakeholders whose decision was not enforced, provided they are interested in this type of discrepancy.

A naïve approach to identify decision mismatches would be to evaluate an access request against the global policy and against each user policy, and compare the obtained decisions. In particular, user policies could be stored separately in the PAP; then, the PDP can fetch one policy at the time for the evaluation of the access request. This naïve approach, however, has a number of drawbacks. First, the selective fetching of policies is not supported by most existing XACML implementations; they typically fetch all policies available in the PAP and then combine the decisions obtained evaluating the fetched policies using a root combining algorithm [14]. Therefore, this approach would requires a modification of existing XACML implementations. In addition, it requires instantiating the PDP for each user policy, affecting performance.

To address these drawbacks, we introduce *viewpoints* to distinguish user policies in the global policy. Every user u submits an XACML policy p_u implementing its authorization requirements. To reflect the viewpoint the target of p_u is extended with an environment attribute `ViewPoint`. Two values are assigned to this attribute: the identifier of u, representing the user viewpoint, and "*global*". The evaluation with respect the *global* perspective provides the access decision which is actually enforced by the

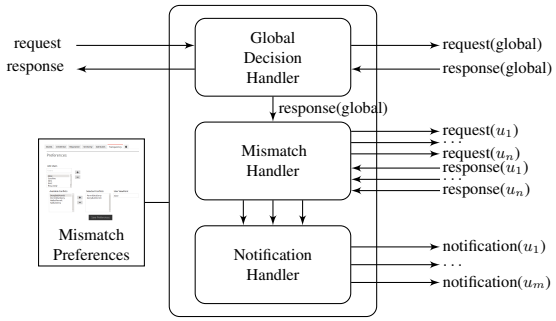


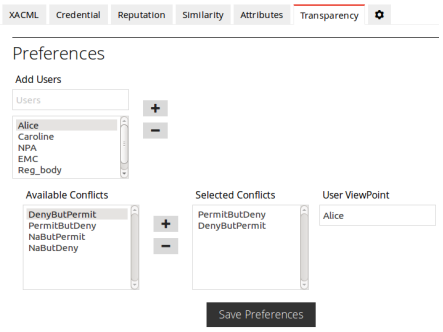
Fig. 3: Transparency Service Architecture

authorization mechanism. It is worth noting that the target is applicable to a given access request (and thus a user policy is evaluated) only if at least one of these two attribute values for attribute `ViewPoint` is provided in the request. User policies are combined based on the role of the corresponding stakeholder with respect to the resource to be protected as described in Section 3. The resulting policy is stored in the PAP and is fetched by the PDP for the evaluation of access requests.

The architecture of the transparency service is presented in Fig. 3. The service comprises three main components: *Global Decision Handler*, *Mismatch Handler* and *Notification Handler*. The service allows users to specify their preferences about which mismatches they want to be notified (e.g., access is permitted whereas the user wants to deny it) along with their contact information. Upon receiving a request, the Global Decision Handler adds attribute `ViewPoint` with value "*global*" to the request and passes on the enriched request for evaluation. The response is passed on for enforcement; it is also sent to the Mismatch Handler. This component checks the mismatch preferences provided by every user to determine the users $u_1, \ldots u_n$ who are interested in mismatches corresponding to the decision reached. For each such user u, the Mismatch Handler creates a new access request which consists of the original request but now extended with attribute `ViewPoint` taking value u, the identifier of the user. As the policies specified by other users will not be applicable (due to a non-matching value for attribute `ViewPoint`) this request is only evaluated against the policy of the corresponding user. When a response to a viewpoint specific requests does not match the global decision and the user is interested in this spefic type of mismatch, the Mismatch Handler calls the Notification Handler. This component retrieves the contact information of the users from the database and notifies them of the mismatches that occurred.

5.2 Service Implementation and Deployment

We have implemented the transparency service within the SAFAX framework [10]. SAFAX is an XACML-based architectural framework that offers authorization as a service. A main characteristic of SAFAX is that all components in the XACML reference architecture are designed as loosely coupled services. These services communicate with each other in JSON or XML via preregistered interfaces (defined in a service registry). SAFAX has been implemented in Java and runs on Apache Tomcat server using Jersey as a service framework. Back-end persistent data are stored in a MySQL server.



(a) Mismatch Preferences

```
RESPONSE for Global
<Response xmlns:ns2="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os">
    <Result>
        <Decision>Permit</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
    </Result>
</Response>
RESPONSE for Alice
<Response xmlns:ns2="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os">
    <Result>
        <Decision>Deny</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
    </Result>
</Response>
NOTIFICATIONS
User Alice had Deny but the response was Permit
```

(b) Response

Fig. 4: SAFAX GUI

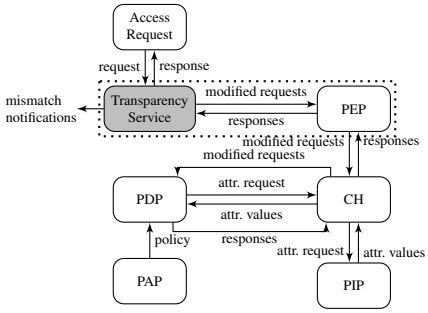


Fig. 5: Transparency Service as PEP

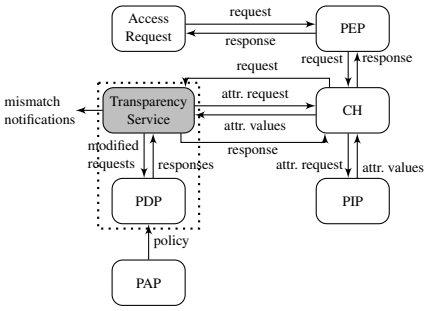


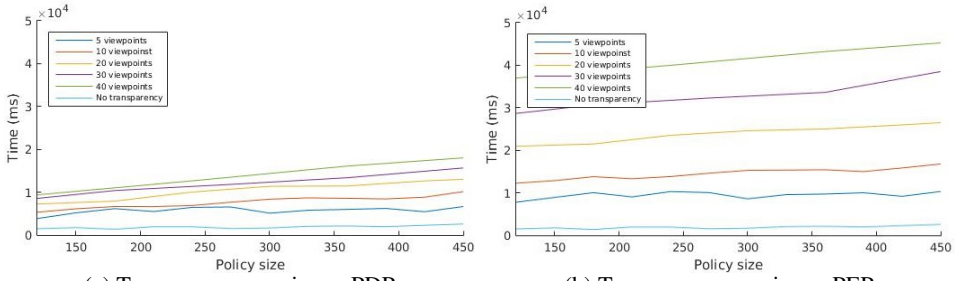
Fig. 6: Transparency Service as PDP

To manage the authorization service configuration and policies, SAFAX offers a User Interface (referred to as SAFAX GUI) that consumes SAFAX services.

The transparency service has been implemented as a SAFAX service and the SAFAX GUI has been extended to manage its configurations. Fig. 4a shows a screenshot of the interface used to manage viewpoints and set stakeholders' mismatch preferences. These preferences are stored in a persistent database on the MySQL server and used by the Notification Handler to determine, for each stakeholder, which mismatches should be notified. For demonstration purposes, the evaluation outcome for every request and the notified mismatches are shown in the SAFAX GUI (Fig. 4b).

Thanks to the service-oriented nature of SAFAX, the transparency service can be deployed at two different locations within the XACML reference architecture. In particular, it can act as either PEP or PDP. Depending on its use, the transparency service and its interfaces have to be registered in the SAFAX service registry accordingly. As shown by the architectures in Fig. 5 and Fig. 6, the transparency service encapsulates rather than replacing the corresponding components. By creating a dependency between the transparency service and one of the existing PEP and PDP services, the expected message flow for the corresponding configuration is achieved.

When the transparency service is used as PEP (Fig. 5), it can be seen as an external service offered to users by a (possibly) different provider. On the other hand, when the



(a) Transparency service as PDP (b) Transparency service as PEP

Fig. 7: Evaluation of the overhead introduced by the transparency service

transparency service is used as PDP (Fig. 6), it can be seen as additional functionality offered by the authorization service itself. SAFAX is able to support both configurations without the need of modifying existing components due to its service-oriented nature. In contrast, other existing XACML implementations can only support the transparency service as an external service because they implement the XACML reference architecture as a monolithic component. Deployment of the transparency service as the PDP would require a modification of these XACML implementations.

6 Evaluation

As discussed in the previous section, the transparency service generates multiple requests to identify mismatches between the decision enforced by the system and user policies. Therefore, we need to evaluate the introduced overhead to ensure it does not affect user experience, thus hampering the adoption of the service in existing infrastructures. For the experiments we created a dataset consisting of policies of different size, where the size of a policy is characterized by the number of rules in the policy. Since the number of generated requests depends on the number of viewpoints, we added a varying number of viewpoints to these policies (i.e., 5, 10, 20, 30 and 40 viewpoints). The same policies were evaluated when the transparency service is deployed as PEP and PDP as well as when the transparency service is not used. We computed the average evaluation time over ten runs; a new policy dataset was created for each run.

The results of the experiments are shown in Fig. 7. These graphs show that the transparency service when deployed as PEP (Fig. 7b) introduces a larger overhead than when it is deployed as PDP (Fig. 7a). When the transparency service is deployed as PEP, every generated request has to be handled by the PEP, CH and PDP (see Fig. 5). On the other hand, when the service is deployed as PDP, requests are only handled by PDP (see Fig. 6), thus leading to a lower overhead. In addition to the deployment method, the results show that the evaluation time depends on the number of viewpoints and policy size. In particular, the number of viewpoints has an impact on the number of requests that are generated. The observed results imply that the delay introduced by the communication among the components of the system is more significant than the overhead due to the evaluation of the requests.

Although enabling transparency unavoidably comes at the cost of computation time, it should be noted that the decision enforced by the authorization mechanism is obtained from the evaluation of the request with the 'global' viewpoint. The other requests are only needed to detect policy mismatches and generate notifications. Therefore, they can be generated and evaluated offline to not affect the functioning of the system.

7 Related Work

With the growing popularity of collaborative systems, the risks of data breaches have increased due to the intrinsic difficulty of establishing a data government model for such systems. Several mechanisms have been proposed to balance the ease of collaboration and the level of security with collaborative systems (see [21] for a survey). For instance, solutions such as Role-Based Access Control [16], Task-Based Access Control [20] and Team-Based Access Control [19] use the roles within an organization, the purpose of the usage or group membership to regulate the access to sensitive data. While these solutions provide some basic features to enable access control in collaborative systems, they usually assume that data objects are under the control of a single entity and, thus, they lack support for policy administration of shared resources.

A few models have been proposed for collaborative authorization management of shared data. For instance, Squicciarini et al. [18] consider resources co-owned by multiple users who can separately specify their policies for the shared data, and use the Clarke-Tax model for the collective enforcement of these policies. Hu et al. [7] propose a multiparty access control model where, in addition to the owner of data, other controllers (e.g., contributor, dissiminator, stakeholders) can regulate the access to shared data. The owner of the data can choose an appropriate strategy (e.g., owner-overrides, full-consensus-permit, majority-permit) to resolve policy conflicts. To account for the different level of authority, the model uses a voting scheme that allows the specification of different weights for controllers. Similarly to the model proposed in [7], our governance model uses policy combination strategies for conflict resolution; however, our model allows a more fine-grained governance of shared resources by representing and ordering levels of authority through an archetype hierarchy that can be instantiated using an arbitrary combination of policy combining algorithms.

Policy combination strategies are often used by authorization mechanisms to define how policy conflicts should be resolved. Examples of conflict resolution strategies are: deny takes precedence [8], permit takes precedence [8], most-specific takes precedence [8, 12] and explicit specification of priorities [17]. Similarly, Reeder et al. [15] propose specificity precedence, deny precedence, order precedence, recency precedence or the combination of these when a single strategy fails. The most prominent authorization mechanism that supports (most of) these strategies is XACML [14]. In particular, XACML encodes conflict resolution strategies as policy combining algorithms (see Section 2). Our solution, being based on XACML, natively supports these strategies as well. Moreover, given the extensible nature of XACML, accommodating other conflict resolution strategies as the ones proposed in [8, 11, 12] is straightforward.

Mazzoleni et al. [13] argue that policy combination algorithms provided by XACML, and in general conflict resolution strategies, are not enough to integrate policies spec-

ified by autonomous parties. To this end, they define a policy similarity process and a number of policy integration algorithms. The policy similarity process is used to analyze the behavior of policies with respect to access requests. The result of this analysis, along with policy integration preferences given by the users, is used to select the policy integration algorithms for building the global policy. Differently from [13], our framework integrates policies specified by multiple administration entities based on their relation with a data object, thus reflecting the level of authority that these entities have on the object. The policy similarity process and policy integration algorithms proposed in [13] can be employed in our framework to form archetype and level policies.

Although methods for integrating policies specified by autonomous entities (as well as conflict resolution strategies) are necessary to ensure the proper functioning of the system, their application makes access decision making non-transparent to users. Transparency has become a major demand for modern IT governance, social and medical systems [1, 4, 9]. However, very little research has been conducted towards its introduction into access control. To the best of our knowledge, CollAC [2] is the only work that proposes a transparent access control solution which detects conflicts during policy evaluation and notifies the users whose decisions have been overridden. This work extents [2] along two main directions. First, this work introduces archetype hierarchies to reason about the level of authority that users have over shared objects together with a method for obtaining the global policy from the archetype hierarchy and user policies. Moreover, we demonstrate how the notion of transparency can be accommodated in existing XACML-based access control mechanisms, thus showing its practical applicability.

8 Conclusion

This paper has introduced a governance model for collaborative systems, which enables the integration of the access requirements of all entities involved in the protection of a data object with respect to their relation with the object. This way, all entities are offered an appropriate level of control over shared resources. We have implemented the model in XACML, allowing each user to provide its requirements as a policy and using appropriate combining algorithms to achieve the right precedence between their policies.

Even if the use of combining algorithms is necessary to automatically resolve policy conflicts and thus guarantee the proper functioning of the system, it can result in a user's policy to be overruled without the user being aware. This may lower the user's trust in the system. To this end, we have introduced transparency in the decision making, allowing users to choose to be notified about conflicts between their access requirements and the decision enforced by the system. Our implementation within SAFAX shows that a transparency service can be deployed both as a PEP and a PDP. Our experiments show that deployment as a PDP has a lower overhead. While the solution is not optimized for performance, it can be applied to many scenarios, especially given the fact that the introduced overhead is not on the critical path for access to resources.

The proposed transparency service only notifies users about policy mismatches. To enhance user awareness, users should be also able to understand why a certain decision was taken [5]. An interesting direction for future work is to augment users' notification with information explaining why their policies were overridden.

Acknowledgments: This work has been partially funded by the ITEA2 projects FedSS (No. 11009) and M2MGrid (No. 13011), the EDA project IN4STARS2.0, and the Dutch national program COMMIT under the THeCS project.

References

1. Albrecht, U.V.: Transparency of health-apps for trust and decision making. Journal of medical Internet research 15(12), e277 (2013)
2. Damen, S., den Hartog, J., Zannone, N.: CollAC: Collaborative access control. In: Proc. of CTS. pp. 142–149. IEEE (2014)
3. Damen, S., Zannone, N.: Privacy implications of privacy settings and tagging in facebook. In: Secure Data Management. pp. 121–138. LNCS 8425, Springer (2013)
4. de Fine Licht, J.: Transparency actually: how transparency affects public perceptions of political decision-making. European political science review 6(02), 309–330 (2014)
5. Ghai, S.K., Nigam, P., Kumaraguru, P.: Cue: A Framework for Generating Meaningful Feedback in XACML. In: Proc. of SafeConfig. pp. 9–16. ACM (2010)
6. Guarda, P., Zannone, N.: Towards the development of privacy-aware systems. Information & Software Technology 51(2), 337–350 (2009)
7. Hu, H., Ahn, G.J., Jorgensen, J.: Multiparty Access Control for Online Social Networks: Model and Mechanisms. TKDE 25(7), 1614–1627 (2013)
8. Jajodia, S., Samarati, P., Sapino, M.L., Subrahmanian, V.S.: Flexible Support for Multiple Access Control Policies. ACM Trans. Database Syst. 26(2), 214–260 (2001)
9. Joshi, A., Bollen, L., Hassink, H.: An empirical assessment of it governance transparency: Evidence from commercial banking. Inf. Sys. Manag. 30(2), 116–136 (2013)
10. Kaluvuri, S.P., Egner, A.I., den Hartog, J., Zannone, N.: SAFAX – An Extensible Authorization Service for Cloud Environments. Frontiers in ICT 2(9) (2015)
11. Li, N., Wang, Q., Qardaji, W., Bertino, E., Rao, P., Lobo, J., Lin, D.: Access control policy combining: Theory meets practice. In: Proc. of SACMAT. pp. 135–144. ACM (2009)
12. Matteucci, I., Mori, P., Petrocchi, M.: Prioritized execution of privacy policies. In: DPM/SETOP. pp. 133–145. LNCS 7731, Springer (2012)
13. Mazzoleni, P., Crispo, B., Sivasubramanian, S., Bertino, E.: XACML Policy Integration Algorithms. ACM Trans. Inf. Syst. Secur. 11(1), 4:1–4:29 (2008)
14. OASIS XACML Technical Committee: eXtensible Access Control Markup Language (XACML) Version 2.0 (2005)
15. Reeder, R.W., Bauer, L., Cranor, L.F., Reiter, M.K., Vaniea, K.: Effects of access-control policy conflict-resolution methods on policy-authoring usability. CyLab p. 12 (2009)
16. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer 29(2), 38–47 (1996)
17. Shen, H., Dewan, P.: Access control for collaborative environments. In: Proceedings of Conference on Computer-supported Cooperative Work. pp. 51–58. ACM (1992)
18. Squicciarini, A.C., Shehab, M., Paci, F.: Collective privacy management in social networks. In: Proc. of WWW. pp. 521–530. ACM (2009)
19. Thomas, R.K.: Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Controls in Collaborative Environments. In: Proc. of RBAC. pp. 13–19. ACM (1997)
20. Thomas, R.K., Sandhu, R.S.: Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management. In: DBSec. pp. 166–181. Springer (1997)
21. Tolone, W., Ahn, G.J., Pai, T., Hong, S.P.: Access control in collaborative systems. ACM Computing Surveys 37(1), 29–41 (2005)