CollAC: Collaborative Access Control

Stan Damen, Jerry den Hartog, Nicola Zannone
Eindhoven University of Technology,
Eindhoven, The Netherlands
s.damen@student.tue.nl,{j.d.hartog,n.zannone}@tue.nl

***Abstract*—Recent years have seen an increasing number of collaborative systems and platforms available online. As the importance of online collaboration grows, so does the need to protect the information used in these systems. In collaborative environments different users may be related to the information in different capacities. This means that traditional access control mechanisms are usually not suitable for collaborative environments as they assume that a single entity is in control of information. Moreover, when different entities can concurrently specify policies for the same resources, the decision making process is not transparent to the users who expect their policies to be enforced by the system. In this paper, we introduce a novel access control framework for collaborative systems. The framework is based on a notion of control which goes beyond the notion of ownership by accounting for the relation of a user with an object. We also make access control decisions transparent by showing where and why collaborative decisions deviate from the policies of single users.**

Keywords*—*access control, collaborative systems, data ownership, transparency, decision mismatch

I. Introduction

In recent years, collaboration has become an important aspect of IT systems. With more and more users connected through the Internet, information creation and management is moving to online collaborative environments. This has spurred the development of collaborative systems such as web application platforms (e.g., Microsoft Sharepoint) and social networks (e.g., Facebook, Google+). Collaborative systems provide users an environment in which they can work together to achieve a common goal [1], [2]. A key feature of collaborative systems is the possibility to share information [3] to achieve this common goal. However, the information might be sensitive and should be accessible only to authorized users.

Information is usually protected by means of access control. Traditional access control systems, however, are not able to deal with the complexity characterizing collaborative environments. In traditional access control systems, control of information is often limited to a single entity (e.g., the system or the owner). In contrast, in collaborative environments information is often created and maintained by several users in different capacities. For instance, information can be provided by a user (data provider), can be stored by some (other) user (data host), and can refer to some (possibly different) users (data subject). Each of these users should have some level of control on the information. To this end, it is necessary to determine the relation a user has with the information to be protected, and thus the associated level of control.

Sharing control between several users also makes it difficult to protect information from unauthorized or accidental disclosure as permissions can be granted to the wrong users [4]. Therefore, fine-grained specification of policies in which exceptions are made explicit must be supported. The concept of exception leads to the differentiation between positive and negative authorization rules [5], where positive rules represent privileges and negative rules represent restrictions.

Systems using both positive and negative authorization rules demand conflict resolution techniques, i.e., strategies that specify how to handle situations where both positive and negative rules apply. Several strategies for policy conflict resolution have been proposed in the literature [6], [7]. Although the use of these strategies is necessary to obtain a definitive decision, either permit or deny, it has an impact on the users' perception towards the system.

Ideally, the system should enforce the policies of all users involved with the management of the information. However, this is not always possible, for instance when users define conflicting policies for the same data object. As a result, it is not easy for users to understand to what extent they can control their information. Resulting lack of confidence in the enforcement of their policies may lead users to not share their data, which would reduce the effectiveness of collaborative systems. In addition, data protection regulations impose that data subjects are in control of how their data are used including who can access them [8]. Therefore, we argue that one of the desiderata of collaborative access control is that the decision making process is transparent to users.

In this paper we present CollAC, a novel access control framework specifically tailored to collaborative environments. In particular, CollAC addressed the issues discussed above by (i) introducing a notion of control which goes beyond the notion of ownership by accounting for the relation of a user with an object; and (ii) enabling transparency in access decision-making through a feedback mechanism.

For the specification and evaluation of collaborative access control policies, we rely on hybrid logic [9]. Bruns et al. [10] have shown that (a fragment of) hybrid logic can be used for the specification of access control policies. The first key feature of hybrid logic is the ability of explicitly specifying the viewpoint of the user from which policies should be evaluated. In addition, this formalism uses the relations between users as a central concept in the decision-making process. In this paper, we take a step further and show how the notion of viewpoint introduced in [10] can be extended to represent the capacity

$$
\begin{array}{ll}
t & ::= n \mid x \\
\varphi & ::= t \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle r\rangle\varphi \mid \langle -r\rangle\varphi \mid @_t\varphi \mid \downarrow x\varphi
\end{array}
$$

Figure 1: Syntax of hybrid logic HL, with $n \in Nom$, $x \in Var$, $p \in P$ and $r \in R$.

of users to control access to information.

When access to information is controlled by different users, it may be impossible to enforce the policies of all users related to the information, especially if they have specified conflicting policies. Nonetheless, users should be aware whether their policies have actually been enforced. To this end, we introduce the notion of transparency in access control. In particular, we enable transparency in CollAC by introducing a feedback system. The policies defined by users are analyzed against the decision made by the system to identify mismatches. Identified mismatches are reported to the users whose policies have not been enforced. Those users can revise their policies based on the received feedback. We demonstrate the proposed access control framework by applying it to a social network setting.

The paper is structured as follows. The next section introduces preliminaries on hybrid logic. Section III introduces the basic components of CollAC, and Section IV presents an architecture for policy evaluation. Section V presents an evaluation of the proposed framework. Finally, Section VI discusses related work, and Section VII concludes the paper and gives directions for future work.

II. Preliminaries

This section presents an overview of hybrid logic and its application for the specification of access control policies.

A. Hybrid logic

Hybrid logics [9] are an extension of modal logic. Their distinguishing concept is the one of *nominal*. Nominals are propositional symbols that are true at exactly one possible world. Therefore, a nominal can be considered as a term referring to a world. To formalize a statement being true at a particular possible world, hybrid logic uses the *satisfaction operator* (denoted by $@_a$). Intuitively, the satisfaction operator determines the "viewpoint" with respect to which a formula is true. Here, we consider the hybrid logic HL proposed in [10].

The syntax of HL is based on four countable and pair wise disjoint sets: the set of nominal symbols *Nom*, the set of variable symbols *Var*, the set of propositional symbols *P*, and the set of relations *Rel*. Formulas of hybrid logic are generated by the grammar in Fig. 1. Intuitively, a term t is either a variable x or a nominal n. A formula is a term, a predicate p, or can be build using operators. The negation ($\neg$) and conjunction ($\wedge$) operators are standard. Operator $\langle r\rangle\varphi$ denotes the possibility modality for relation r ($\langle -r\rangle\varphi$ is similar, but the relation is inverted). The satisfaction operator $@_t\varphi$ expresses that φ is true in the world to which t refers. The binder operator $\downarrow x\varphi$ binds the current world to variable x. By binding a variable to a specific world, it is possible to create a name for the current world and use such a name later in the formula.

$$
\begin{array}{lcl}
M, w, g \models x & \stackrel{\text{def}}{=} & w = g(x) \\
M, w, g \models n & \stackrel{\text{def}}{=} & V(n) = \{w\} \\
M, w, g \models p & \stackrel{\text{def}}{=} & w \in V(p) \\
M, w, g \models \neg\varphi & \stackrel{\text{def}}{=} & M, w, g \not\models \varphi \\
M, w, g \models \varphi_1 \wedge \varphi_2 & \stackrel{\text{def}}{=} & M, w, g \models \varphi_1 \text{ and } M, w, g \models \varphi_2 \\
M, w, g \models \langle r\rangle\varphi & \stackrel{\text{def}}{=} & M, w', g \models \varphi \text{ for some } (w, w') \in \Omega(r) \\
M, w, g \models \langle -r\rangle\varphi & \stackrel{\text{def}}{=} & M, w', g \models \varphi \text{ for some } (w', w) \in \Omega(r) \\
M, w, g \models @_n\varphi & \stackrel{\text{def}}{=} & M, w^*, g \models \varphi \text{ where } V(n) = w^* \\
M, w, g \models @_x\varphi & \stackrel{\text{def}}{=} & M, g(x), g \models \varphi \\
M, w, g \models \downarrow_x \varphi & \stackrel{\text{def}}{=} & M, w, g[x \mapsto w] \models \varphi \\
 & & \\
M, g \models \varphi & \textit{iff} & \forall w \in M \;\; M, w, g \models \varphi \\
M, g \not\models \varphi & \textit{iff} & \forall w \in M \;\; M, w, g \not\models \varphi
\end{array}
$$

Figure 2: Satisfaction relation $M, w, g \models \varphi$ specifying that formula φ is true in world w of model M under valuation g.

Disjunction ($\vee$), implication ($\rightarrow$) and necessity modality ($[r]$) can be defined using the above operators in the standard way.

A *model* M of HL is a triple (W, Ω, V) where W is a non-empty set of possible worlds, $\Omega : Rel \rightarrow (W \rightarrow 2^W)$ such that, for all $r \in Rel$, $\Omega(r)$ is a binary relation on W, and $V : Nom \cup P \rightarrow 2^W$ a total function with $V(n)$ a singleton for all $n \in Nom$. An *instantiation* $g : Var \rightarrow W$ is a total function from variables to worlds. We write $g[x \mapsto w]$ for the instantiation that maps x to w, and maps y to $g(y)$ if $y \neq x$.

The semantics of HL formulas is defined by the satisfaction relation $M, w, g \models \varphi$ (Fig. 2). Nominals and variables are true at a single world. Their semantics are defined through functions V and g, respectively. Propositions are true at zero, one or more worlds. The semantics of negation and conjunction are standard. The possibility modality $\langle r\rangle\varphi$ is true at world w if relation r holds between w and w', and φ is true at world w'; the semantics of $\langle -r\rangle\varphi$ is dual. The satisfaction statement $@_t\varphi$ is true if φ is true at the unique world identified by t with respect to functions V (if t is a nominal) or g (if t is a variable). Formula $\downarrow_x \varphi$ is true at world w if φ is true at w where g is updated so that x identifies w. We write $M, g \models \varphi$ and $M, g \not\models \varphi$ to indicate that $M, w, g \models \varphi$ is true at all world w and false at all worlds w, respectively.

B. Hybrid logic as a policy language

Bruns et al. [10] propose to use a fragment of HL as a language for the specification of relationship-based access-control policies. To this end, they introduce the notion of *protection state*. A protection state is a model M of HL in which the elements of W denote users. Users are related to objects in the system as *owners* or *requesters*. HL formulas express access control policies from the viewpoint of single users. The set of relations Rel is used to specify the relationships between users upon which access decisions are based. The fragment of HL used for the specification of access control policies is formally defined as follows [10].

Definition 1. *Let* HL(**own**,**req**) *be the set of HL formulas that*

- *contain at most **own** and **req** as free variable.*
- *are Boolean combinations of formulas of the form $@_{own}\varphi$ and $@_{req}\varphi$.*

Policies are formulas of HL(**own**,**req**), which have the form

$$@_{own}\varphi_1 \wedge @_{req}\varphi_2$$

where $@_{own}\varphi_1$ denotes the viewpoint of the owner, and $@_{req}\varphi_2$ the viewpoint of the requester.

Example 1. *A user wants to restrict the access to an object she owns to her friends and colleagues. This policy can be expressed as follows*

$$@_{own}(\langle friend\rangle \mathbf{req} \vee \langle colleague\rangle \mathbf{req})$$

Access decisions are made by evaluating access requests against access control policies. Given a protection state M, a user u_1 has permission to access an object owned by user u_2 according to a policy pol in HL(**own**,**req**) iff $M, [\mathbf{own} \mapsto u_2, \mathbf{req} \mapsto u_1] \models pol$. Intuitively, the free variables **own** and **req** are bound to the owner of object u_2 and to the requester u_1, respectively. Then, the HL formula is evaluated with respect to those bindings.

III. CollAC

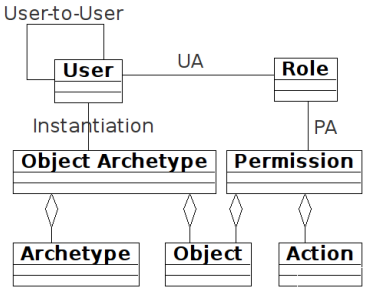
Although HL encompasses features suitable for the specification and enforcement of collaborative access control policies, it imposes restrictions that make it inapplicable in practice. First, it assumes that data are owned by single users. The data owner solely is responsible for defining the security policy regulating the access to her data. This view, however, is too limited for modern collaborative systems. Data objects have multiple users related to them in different capacities. Depending on the relation of a user with an object, the user should retain some authority on the object [11]. In addition, HL only allows the specification of positive authorizations. When the policies of more than one user are used to make a decision, a finer-grained access control model able to deal with exceptions is required [5].

In this section, we introduce CollAC, a collaborative access control model that extends HL by addressing the limitations above. In particular, CollAC allows multiple users to define security policies for the same data object and makes it possible to specify the users from whose viewpoint security policies should be evaluated. Moreover, CollAC allows the specification of negative authorizations to enable a better control of data in collaborative systems. First, we present the conceptual model of CollAC. Then, we define the syntax and semantics of authorization policies and policy evaluation. Finally, we introduce the notion of transparency for access control.

A. Conceptual Model

CollAC is based on role-based access control (RBAC) and on the policy language in [10]. From RBAC, it employs the notion of user, role, permission, and user and permission assignment. From [10], it employs constructs to represent relations between users. The CollAC metamodel along with its formal specification is presented in Fig. 3.

A *user* is an active entity that can exercise permissions on objects. Relations between users are captured by relation *user-to-user* which defines how users are connected, e.g. friends,



(a) CollAC metamodel

Element	**Specification**
Users	U
Objects	O
Actions	A
Roles	R
Archetypes	AT
Object Archetypes	$OAT \subseteq AT \times O$
Permissions	$PER \subseteq O \times A$
Relations	
Permission assignment	$PA : R \to 2^{PER}$
User assignment	$UA : U \to 2^R$
User-to-User	$\Omega(r) : U \to 2^U$ s.t. $r \in Rel$
Instantiation	$g : OAT \to U$

(b) CollAC formal specification

Figure 3: CollAC model

colleagues. In particular, *user-to-user* models the family of relations $\Omega(r)$ (Section II-A). Objects are entities to be protected; *actions* are operations that can be performed on objects, e.g. delete or read. As in RBAC, *permissions* are represented as pairs of actions and objects. A *role* is the abstract characterization of the behavior of an active entity. Users are assigned to roles through *user assignments* (*UA*), and permissions to roles through *permission assignments* (*PA*).

To characterize the relation between users and objects, we introduce the notion of archetype. Intuitively, an *archetype* extends the notion of owner and requester in [10] by representing the type of capacity in which a user can be related to an object, e.g. data host, data provider and requester. An *object archetype* represents this capacity with respect to a given object. Object archetypes are associated to users using relation *instantiation*. This relation corresponds to instantiation g in Section II-A.

B. Syntax and Semantics

Before defining the CollAC policy language, we introduce some notation. Given a formula φ in HL, $\tau(\varphi)$ denotes the set of free variables in φ.

Definition 2. *An* authorization statement δ *is a formula in* $HL(\textbf{\textit{Subj}})$ *of the form* $@_t\varphi$ *where*

- $t \in U \cup OAT$ *represents the subject for whom the statement is specified,*
- $\tau(\delta) \subseteq \textbf{\textit{Subj}}$ *with* $\textbf{\textit{Subj}} \subseteq OAT$.

Definition 3. *An* authorization rule *is a logical combination of authorization statements of the form:*

$$\delta_1 \circ \delta_2 \circ \cdots \circ \delta_n$$

where δ_i is (a reference to) an authorization statement and $\circ \in \{\wedge, \vee\}$.

Definition 4. *An* authorization policy *is a pair $\langle \phi, \psi \rangle$, where ϕ is a positive authorization rule and ψ is a negative authorization rule.*

Example 2. *A social network defines default policies for the data objects posted by users on the social network site. These policies are defined on behalf of the users related to an object: the* data host *(DH), i.e. the user in whose profile the object is posted; the* data provider *(DP), i.e. the user who uploaded the object; and the* data subject *(DS), i.e. the user to whom the object refers. For photos, the social network states that users who are (i) friends or relatives of the data host and (ii) friends of the data provider can view the photo. However, (iii) the relatives of the data subject cannot view the photo. The above positive and negative authorizations can be expressed as authorization statements:*

(i) $\quad @_{DH}(\langle friend \rangle \mathbf{req} \vee \langle relative \rangle \mathbf{req})$
(ii) $\quad @_{DP}\langle friend \rangle \mathbf{req}$
(iii) $\quad @_{DS}\langle relative \rangle \mathbf{req}$

The overall policy is obtained by combining the above authorization statements. In particular, it is the pair $\langle \phi, \psi \rangle$ where the positive authorization rule ϕ and negative authorization rule ψ are defined as follows:

$$\phi = @_{DH}(\langle friend \rangle \mathbf{req} \vee \langle relative \rangle \mathbf{req}) \wedge @_{DP}\langle friend \rangle \mathbf{req}$$
$$\psi = @_{DS}\langle relative \rangle \mathbf{req}$$

C. Policy Evaluation

When an access request is received, authorization policies have to be instantiated on the basis of the requested object. To keep the notation simple, we consider that the instantiation used to assess the satisfaction relation is the one induced by the access request, i.e. it contains the variable assignments induced by the request. The access decision is made by evaluating the positive and negative authorization rules individually and then combining the results. In this section, we formally define policy evaluation. First, we introduce a rule evaluation function to evaluate authorization rules; then we define a policy evaluation function that computes an access decision based on the evaluation of positive and negative authorization rules.

Definition 5. *Let M be a model of HL, AR the set of authorization rules, φ be an authorization rule in AR, and g the instantiation induced by an access request. The* rule evaluation function *is a function $\mathbf{E} : AR \to \{0, 1\}$ s.t.*

$$\mathbf{E}(\varphi) = \begin{cases} 1 & \text{if } M, g \models \varphi \\ 0 & \text{if } M, g \not\models \varphi \end{cases}$$

where 1 represents that the rule is applicable to the access request in M, and 0 that the rule is not applicable to the access request in M.

Note that authorization rules always define the perspective (or worlds) from which formulas are evaluated; thus $M, w, g \models \varphi$ iff $M, g \models \varphi$. The evaluations of the positive and negative rules have to be combined to determine the access decision for the request.

Definition 6. *Let AR be the set of authorization rules, and $\langle \phi, \psi \rangle$ an authorization policy s.t. $\phi, \psi \in AR$. The* policy evaluation function *is a function $\mathbf{F} : AR \times AR \to \{P, D, N/A, C\}$ s.t.*

$$\mathbf{F}(\phi, \psi) = \begin{cases} P & \text{if } \mathbf{E}(\phi) = 1 \wedge \mathbf{E}(\psi) = 0 \\ D & \text{if } \mathbf{E}(\phi) = 0 \wedge \mathbf{E}(\psi) = 1 \\ N/A & \text{if } \mathbf{E}(\phi) = 0 \wedge \mathbf{E}(\psi) = 0 \\ C & \text{if } \mathbf{E}(\phi) = 1 \wedge \mathbf{E}(\psi) = 1 \end{cases}$$

where P represents permit*, D* deny*, N/A* not applicable*, and C* conflict*.*

Intuitively, the policy evaluation function uses the rule evaluation function to evaluate the positive and negative rules forming the policy individually and then combines the results. In particular, it returns decision permit (P) if only the positive rule is applicable. Similarly, it returns decision deny (D) if only the negative rule is applicable. If both positive and negative rules are not applicable, the decision is not applicable (N/A). A conflict (C) is identified when both the positive and negative rules are applicable; indeed, in this case a definite decision, either permit or deny, cannot be made.

Example 3. *Consider policy $\langle \phi, \psi \rangle$ in Example 2. Eve wants to see a photo about Charlie posted in Alice's profile. The photo was posted by Bob. Eve is friend of both Alice and Bob; she is also Charlie's relative. The policy is thus instantiated with respect to the viewpoint of the data host (Alice), data provider (Bob) and data subject (Charlie) of that photo. Based on the relation Eve has with those users, we obtain*
$M, g[DH \mapsto alice, DP \mapsto bob, DS \mapsto charlie, \mathbf{req} \mapsto eve] \models \phi$
$M, g[DH \mapsto alice, DP \mapsto bob, DS \mapsto charlie, \mathbf{req} \mapsto eve] \models \psi.$
Therefore, $\mathbf{F}(\mathbf{E}(\phi), \mathbf{E}(\psi)) = F(1, 1) = C$.

The access control mechanism has to enforce the decision made within policy evaluation. However, only definitive decisions – permit and deny – can be directly enforced; non-definitive decisions – not applicable and conflict – have to be interpreted before they can be enforced. To this end, we introduce a decision resolution function which maps non-definitive decisions to definitive decisions.

Definition 7. *Let AR be the set of authorization rules, $\langle \phi, \psi \rangle$ an authorization policy s.t. $\phi, \psi \in AR$. A* decision resolution function *is a function $\mathbf{DR} : AR \times AR \to \{P, D\}$ s.t.*

$$\mathbf{DR}(\phi, \psi) = \begin{cases} \mathbf{F}(\phi, \psi) & \text{if } \mathbf{F}(\phi, \psi) \in \{P, D\} \\ \mathbf{S}(\mathbf{F}(\phi, \psi)) & \text{if } \mathbf{F}(\phi, \psi) = \{N/A, C\} \end{cases}$$

where function $\mathbf{S} : \{N/A, C\} \to \{P, D\}$ maps a non-definitive decision into a definitive decision.

Different strategies can be employed to obtain a definitive decision. For instance, two well established approaches to reduce non-definitive decisions into definitive decisions have been proposed in [12]: *permit-takes-precedence* which maps

non-definitive decisions to permit, and *deny-takes-precedence* which maps non-definitive decisions to deny.

One may argue that the final decision should reflect the policy of the data subject. However, 'control by data subjects' does not necessarily mean that data subject decision always prevails. For example, different data subjects may specify conflicting requirements. Also, confidentiality of data may deny access to data for which the data subject has consented. On the other hand, access to data may need to be given even if the data subject does not consent (e.g., access by an investigator to a criminal record, a doctor to a medical file in emergency situations).

Example 4. *Consider the policy evaluation in Example 3. Here, the authorization decision obtained using the policy evaluation function is* conflict. *If a deny-takes-precedence strategy is used to obtain a definitive decision, the final authorization decision is* deny *and thus Eve is not allowed to view the photo.*

D. Transparency

Ideally, the policies of all the users related to the requested object should be enforced. This, however, is often not possible. The decision resolution function (Def. 7) maps non-definitive decisions to definitive decisions. If both positive and negative authorization rules within a policy are applicable (i.e., the policy evaluation function returns a conflict), one of these rules has to be overruled to make a definitive decision, either permit or deny. In particular, the decision resolution function discards the positive authorization rule if deny-takes-precedence is used and discards the negative authorization rule if permit-takes-precedence is used. As a result, the authorization statements forming the discarded authorization rule are not considered in the final access decision.

Users typically define authorization statements to protect their data, and rely on the system for their enforcement. However, users are often not aware whether their authorization statements have actually been enforced. This lack of confidence may reduce the level of trust users have towards the system. We argue that a desideratum of collaborative access control is that the decision making process is transparent to users. In particular, it should enforce the policies specified by users, and notify and justify the decision made to users when this is not possible. To enable transparency in access control, we introduce the notion of *mismatch*. Intuitively, mismatches are used to indicate that policy evaluation differs from what is intended by a user due, for instance, to decision resolution.

Definition 8. *Let φ be an authorization rule, and Δ^{φ} the set of authorization statements occurring in φ. An* applicability mismatch *occurs if*

$$\exists \delta \in \Delta^{\varphi} \text{ s.t. } \mathbf{E}(\delta) = 1 \wedge \mathbf{E}(\varphi) = 0.$$

Intuitively, an applicability mismatch indicates that the authorization statement of a user is not considered in the access decision because the authorization rule containing the statement is not applicable.

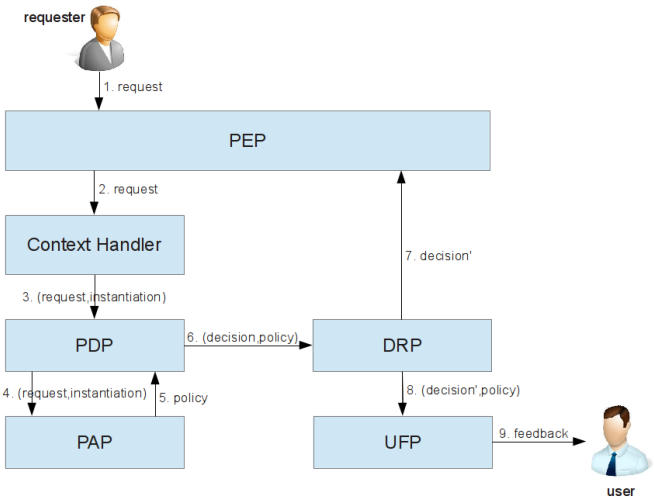


Figure 4: CollAC architecture

Definition 9. *Let $\langle \phi, \psi \rangle$ be an authorization policy, and $\Delta^{\phi}, \Delta^{\psi}$ the sets of authorization statements occurring in ϕ, ψ respectively. A* decision mismatch *occurs if*

- $\exists \delta \in \Delta^{\phi}$ *s.t.* $\mathbf{E}(\delta) = 1 \wedge \mathbf{DR}(\phi, \psi) \neq P$, *or*
- $\exists \delta \in \Delta^{\psi}$ *s.t.* $\mathbf{E}(\delta) = 1 \wedge \mathbf{DR}(\phi, \psi) \neq D$.

Intuitively, a decision mismatch indicates that the final access decision does not match the intended decision of the user. Thus, conflicts are captured by decision mismatches.

Example 5. *Consider policy $\langle \phi, \psi \rangle$ in Example 2 and the instantiation in Example 3. If a deny-takes-precedence strategy is adopted, the conflict is resolved in* deny *(Example 4). Therefore, the final decision is different from what is intended by Alice and Bob, i.e. Alice and Bob have a decision mismatch.*

Now, suppose that Eve is only friend of Alice (and not friend of Bob) and she is Charlie's relative. In this case, policy evaluation results in a deny *decision. Although Alice's authorization statement is applicable and thus she is expecting a* permit *decision, the positive authorization rule is not applicable. Therefore, Alice has a decision mismatch due to an applicability mismatch.*

IV. Architecture

The previous section captures the main ingredients of CollAC, namely policy specification and evaluation as well as the notion of transparency for access control. In this section, we present the CollAC architecture along with the data flow (Fig. 4). The architecture resembles the one of XACML [6].

The architecture consists of the following components. The *policy enforcement point* (PEP) intercepts the authorization requests from users and enforces authorization decisions. The *context handler* is responsible to collect the information necessary for the evaluation of authorization requests.[1] The *policy decision point* (PDP) evaluates authorization requests

[1]In the XACML architecture, the context handler performs this task in collaboration with the policy information point (PIP) [6]. For the sake of simplicity, here we omit the PIP.

against authorization policies. The *policy administration point* (PAP) maintains the database of authorization policies and provides the correct policy to the PDP for the evaluation of authorization requests. The *decision resolution point* (DRP) is responsible to make a definitive decision based on a given decision resolution strategy. The *user feedback point* (UFP) detects mismatches and provides feedback to those users whose authorization statements have not been enforced.

Figure 4 shows the architectural model along with the data flow model. An authorization request is intercepted by the PEP (1). The PEP forwards the request to the context handler (2). The latter checks the request and determines the instantiation based on the identity of the requester and on the requested object. The request along with the instantiation is sent to the PDP (3). The PDP retrieves the appropriate authorization policy by contacting the PAP ((4) and (5)). Then, the PDP uses the policy evaluation function to evaluate the request against the retrieved policy (Section III-C). The decision is sent to the DRP, along with the authorization policy used for the evaluation (6). The DRP maps the decision made by the PDP into a definitive decision based on the specified decision resolution strategy and sends the final decision to the PEP (7). The final decision along with the evaluated policy is also sent to the UFP (8). The UFP detects mismatches and contacts the users whose authorization statements have not been enforced by providing motivations for it (9).

V. Evaluation

This section describes several experiments performed to validate the feasibility of our approach. Although transparency is an important aspect of collaborative systems, users should not be overwhelmed by notifications about their policies not being enforced. We therefore consider the amount of feedback sent to users in a social network scenario. We also check that policy evaluation performance is acceptable.

A. Setting

As shown in [13], social networks often resemble a structure in which there are several clusters; each cluster is close to being fully connected, whereas links between clusters are sparse. For our experiments we generate a sample dataset consisting of 100 users and 100 objects that mimics the structure of a social network. Users are divided into clusters and connected to each other through four types of relations. Within a cluster, users have a substantially higher probability of having a particular relation with other users compared to users outside the cluster. Authorization policies are created by assigning a data host, a data provider and a data subject to each object and generating the necessary user statements and authorization rules. Each user can define his policies stating who is allowed to access objects to which he is related. Positive and negative rules are applicable if the user statements of each user related to the object are satisfied. Note that the scenario represents a more or less worst case scenario with respect to the number of mismatches generated.

We analyze the effect of *policy complexity* and *network sociability* (i.e., the density of relations between users) on performance and on the number of mismatches and thus the amount of feedback that needs to be sent to the users. We vary policy complexity by changing the size (2, 4 or 6 clauses) of each of the six user statements comprising the positive and negative rules. We vary network sociability by creating two variants of the social network: one with a high relation density in which the average number of relations per user is 10.5, and one with a low relation density in which the average number of relations per user is 7.5.

B. Results

In our experiments we evaluate 3000 access requests for each case. Table I and Table II present the average results over five repetitions of the experiment for each case. The first two columns show the size of user statements in the positive and negative rules, respectively. Column 3 shows the number of relations in the case. Columns 4 to 7 show the number of queries mapped to a certain decision before applying the decision resolution function, while columns 8 to 9 shows the number of queries mapped to a certain decision after applying the decision resolution function. Notice that in our experiments we used a deny-takes-precedence strategy; accordingly, the number of deny decision (column 8) after decision resolution is equal to the sum of deny decision (column 5), conflicts (column 6) and not applicable decision (column 7) before decision resolution. Columns 10 and 11 respectively report the number of applicability and decision mismatches. Finally, the last column reports the computation time in seconds.

From Table I and Table II, we can observe that applicability mismatches range between 1.55 and 1.64 per query for the high density network, and decision mismatches range between 1.19 and 1.44 per query. For the low density network, applicability mismatches range between 1.47 and 1.59 per query, and decision mismatches range between 0.87 and 1.19 per query. The amount of feedback to be sent to the user, however, may be lower. When the preliminary decision is permit or deny, applicability and decision mismatches coincide; thus only one feedback should be reported to the user. When the preliminary decision is conflict, only decision mismatches can occur. From a user perspective, these mismatches occur independently from the user statements defined by other users and thus they may not be reported. When the preliminary decision is not applicable, we have to distinguish whether the mismatches occur in a positive or a negative rule. Applicability mismatches occurring in positive rules result in decision mismatches; thus the feedback can be sent only once. On the other hand, only applicability mismatches can occur in negative rules. These mismatches do not have to be reported as the final decision matches the behavior intended by the user.

The number of relations between users has an impact on the number of decision mismatches. In particular, the lower the number of relations, the lower the number of decision mismatches. This is due to the fact that, in low density networks, it is more likely that user statements do not apply. Conversely,

TABLE I: RESULTS FOR HIGH RELATION DENSITY NETWORK

Statements		Relations	Preliminary				Final		Mismatches		Time
positive	negative	(total)	permit	deny	conflict	n/a	permit	deny	applicability	decision	(sec)
2	2	10707	248	278	243	2231	248	2752	4814	3571	35
2	4	10683	194	363	260	2182	194	2806	4912	3595	45
2	6	10624	135	447	320	2098	135	2865	4768	3608	88
4	2	10707	358	192	300	2150	358	2642	4732	3907	48
4	4	10629	288	272	365	2075	288	2712	4718	3969	119
4	6	10595	226	353	428	1994	226	2774	4807	4139	257
6	2	10680	507	124	315	2053	507	2493	4690	4108	72
6	4	10712	357	200	464	1979	357	2643	4663	4331	368
6	6	10607	269	302	491	1938	269	2731	4643	4289	788

TABLE II: RESULTS FOR LOW RELATION DENSITY NETWORK

Statements		Relations	Preliminary				Final		Mismatches		Time
positive	negative	(total)	permit	deny	conflict	n/a	permit	deny	applicability	decision	(sec)
2	2	7527	150	166	57	2627	150	2850	4480	2610	21
2	4	7501	129	280	92	2499	129	2872	4417	2635	23
2	6	7494	113	373	138	2376	113	2887	4513	2803	27
4	2	7518	269	145	98	2490	269	2732	4612	3049	25
4	4	7501	220	212	144	2424	220	2780	4671	3131	33
4	6	7485	185	303	205	2308	185	2815	4772	3309	45
6	2	7415	368	78	115	2439	368	2632	4505	3228	25
6	4	7527	313	169	172	2346	313	2687	4458	3311	33
6	6	7415	264	220	261	2255	264	2736	4672	3579	108

the number of applicability mismatches is similar in the two variants. This is due to the higher number of requests mapped to not applicable in the low density scenario. The influence of policy complexity on applicability mismatches is limited. The number of decision mismatches increases which is due to the construction of the policies in our scenario; more complex user statements on average apply to more users, thus increasing the number of applicable clauses. Overall we can conclude that the number of mismatches should be manageable for the framework but an effective way to present them to the user is needed. In some scenarios the number of feedback messages can be too large to be very informative for a user.

Finally, the overhead introduced by the framework is acceptable. Policy evaluation time increases with higher policy complexity and relation density. The computation time ranges from 7ms (first row in Table II) to 263ms (last row in Table I) per query. It is worth noting that the computation time was mainly required to evaluate the policy in Prolog (compared to the detection of mismatches).

VI. Related Work

The increasing popularity of collaborative systems requires the development of an access control model able to cope with the challenges posed by those systems. Collaborative access control aims to balance the competing goals of collaboration and security [14]: collaborative systems aim to facilitate the sharing of information, while security aims to protect the same information. A number of collaborative access control models have been proposed in the literature [15]–[19]. For instance, Lin et al. [19] propose an access control model for collaborative environments based on the notion of policy decomposition. The main idea is that the global access control policy is decomposed among parties and each party evaluates a portion of the policy. The decisions obtained from the evaluation of the portion of the policy are then combined to derive the access decision for the collaboration as a whole. Giunchiglia et al. [18] propose RelBAC, a new access control model for web applications. Similarly to our approach, access decisions are made on the basis of the relations between users. However, existing collaborative access control models do not consider the relation of users with the information to determine in which capacity users can control access to information.

Typically, access control models assume that information is controlled by a single entity (the system or the owner of the information). The problem of having different authorities which can define policies is addressed in trust management [20]–[23]. A distinguishing ingredient of trust management systems is that the policies of a principal can refer to other principals' policies, thereby delegating authority to them. However, trust management approaches also assume the presence of a single owner; the owner may delegate the definition of who is authorized to access his resource to other principals. In contrast, in our work we study the situation in which different authorities can concurrently specify access control policies regulating the access to the same resources.

The presence of multi authorities which concurrently control access to the same resources demands for conflict resolution techniques. Several strategies to resolve conflicts in policy evaluation have been proposed in the literature [12], [24], [25]. For instance, Reeder et al. [25] propose to use specificy precedence with respect to both principals and resources in order to resolve conflicts when possible and to resort to deny precedence only when specificity precedence fails. Matteucci et al. [24] propose a strategy for policy conflict resolution based on an approach for multi-criteria decision making. In particular, the analytic hierarchy process is used to represent and quantify policy elements with respect to the overall system

goals and to evaluate alternative solutions. Other approaches like XACML [6] and the work in [7] address conflict resolution through policy combining algorithms which define the procedure to combine the individual results obtained by the evaluation of the rules of the policies. These proposals are orthogonal to our work. In particular, existing conflict resolution strategies and combining algorithms can be complemented with the transparency mechanism presented in this work to make users aware why their policies have not been enforced.

The notion of transparency is emerging as a key aspect to build user trust in IT systems [26], [27]. Several solutions such as privacy dashboards [28] and privacy room [29] provide tools (e.g., browser add-ons, mobile applications) for assessing the exposure of user data and thus enabling transparency. Although these proposals aim to increase user awareness, they do not provide users with insight on access decision-making. To the best of our knowledge, this is the first work that addresses the problem of transparency in access control.

VII. Conclusions

In this paper we have presented CollAC, an access control model for collaborative systems based on hybrid logic. CollAC makes it possible to distinguish to what capacity users can control objects and to represent the user viewpoint from which policies should be evaluated. We argue that in collaborative systems it is essential to be able to specify policies and reason from multiple points of view. The core of hybrid logic is dealing with different system views and it has been shown to be applicable to access control. Although other formalism may be able to express this, or be extended to capture such situations, it is an intrinsic part of hybrid logic making it a natural choice. A fundamental requirement for collaborative access control is that the decision making process is transparent to users. To this end, we have introduced the notion of mismatch to indicate when access decisions made by the system differ from what is intended by users.

The work presented in this paper suggests some interesting directions for future work. In this work, we have adopted a rather simple policy specification language. We aim to extend existing standards like XACML with the notion of archetype. This will allow the definition of policy combining algorithms that combine access decisions on the basis of the capacity of users in controlling access to resources (e.g., data-host-overrides, data-subject-overrides). In addition, a transparent decision making process for XACML would benefit individuals' privacy protection. To this end, XACML PDP can be complemented with a component that analyzes the evaluated policies and detect mismatches between the final access decision and the decision intended by individuals.

Acknowledgment

This work has been partially funded by the Dutch national program COMMIT under the THeCS project, the ITEA2 project FedSS, and the ARTEMIS project ACCUS.

References

[1] M. Beyerlein, S. Freedman, C. McGee, and L. Moran, *Beyond teams: building the collaborative organization*. Jossey-Bass/Pfeiffer, 2003.
[2] P. Johnson-Lenz and T. Johnson-Lenz, *Rhythms, Boundaries, and Containers*. Awakening Technology, 1990.
[3] D. Trivellato, N. Zannone, M. Glaundrup, J. Skowronek, and S. Etalle, "A semantic security framework for systems of systems," *Int. J. Cooperative Inf. Syst.*, vol. 22, no. 1, 2013.
[4] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone, "Requirements engineering for trust management: model, methodology, and reasoning," *Int. J. Inf. Sec.*, vol. 5, no. 4, pp. 257–274, 2006.
[5] S. D. C. D. Vimercati, S. Foresti, P. Samarati, and S. Jajodia, "Access control policies and languages," *IJCSE*, vol. 3, no. 2, pp. 94–102, 2007.
[6] "eXtensible Access Control Markup Language (XACML) Version 3.0," OASIS, OASIS Standard, 22 January 2013. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf
[7] P. Mazzoleni, B. Crispo, S. Sivasubramanian, and E. Bertino, "Xacml policy integration algorithms," *TISSEC*, vol. 11, no. 1, 2008.
[8] P. Guarda and N. Zannone, "Towards the development of privacy-aware systems," *Information & Software Technology*, vol. 51, no. 2, pp. 337–350, 2009.
[9] C. Areces and B. t. Cate, "Hybrid logics," *Studies in Logic and Practical Reasoning*, vol. 3, pp. 821–868, 2007.
[10] G. Bruns, P. Fong, I. Siahaan, and M. Huth, "Relationship-based access control: its expression and enforcement through hybrid logic," in *CODASPY*. ACM, 2012, pp. 117–124.
[11] S. Damen and N. Zannone, "Privacy Implications of Privacy Settings and Tagging in Facebook," in *Secure Data Management*. Springer, 2013.
[12] S. Jajodia, P. Samarati, M. Sapino, and V. Subrahmanian, "Flexible support for multiple access control policies," *TODS*, vol. 26, no. 2, pp. 214–260, 2001.
[13] S. Catanese, P. Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Extraction and analysis of facebook friendship relations," in *Computational Social Networks*. Springer, 2012, pp. 291–324.
[14] W. Tolone, G. Ahn, T. Pai, and S. Hong, "Access control in collaborative systems," *ACM Computing Surveys*, vol. 37, no. 1, pp. 29–41, 2005.
[15] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. M. Thuraisingham, "Semantic web-based social network access control," *Computers & Security*, vol. 30, no. 2-3, pp. 108–115, 2011.
[16] E. Cohen, R. Thomas, W. Winsborough, and D. Shands, "Models for coalition-based access control," in *SACMAT*. ACM, 2002, pp. 97–106.
[17] W. K. Edwards, "Policies and roles in collaborative applications," in *CSCW*. ACM, 1996, pp. 11–20.
[18] F. Giunchiglia, R. Zhang, and B. Crispo, "RelBAC: Relation Based Access Control," in *SKG*. IEEE, 2008, pp. 3–11.
[19] D. Lin, P. Rao, E. Bertino, N. Li, and J. Lobo, "Policy decomposition for collaborative access control," in *SACMAT*. ACM, 2008, pp. 103–112.
[20] M. Becker, J. Mackay, and B. Dillaway, "Abductive authorization credential gathering," in *POLICY*. IEEE, 2009, pp. 1–8.
[21] N. Li, W. Winsborough, and J. Mitchell, "Distributed Credential Chain Discovery in Trust Management," *JCS*, vol. 11, no. 1, pp. 35–86, 2003.
[22] D. Trivellato, N. Zannone, and S. Etalle, "GEM: a Distributed Goal Evaluation Algorithm for Trust Management," *Theory and Practice of Logic Programming*, 2014, to appear.
[23] C. Zhang and M. Winslett, "Distributed authorization by multiparty trust negotiation," in *ESORICS*, ser. LNCS. Springer, 2008, pp. 282–299.
[24] I. Matteucci, P. Mori, and M. Petrocchi, "Prioritized execution of privacy policies," in *DPM*, ser. LNCS 7731. Springer, 2013, pp. 133–145.
[25] R. Reeder, L. Bauer, L. Cranor, M. Reiter, and K. Vaniea, "Effects of access-control policy conflict-resolution methods on policy-authoring usability," Carnegie Mellon University, CMU-CyLab-09-006, 2009.
[26] "Building Customer Trust In Cloud Computing With Transparent Security," Sun Microsystems, Inc, White Paper, 2009.
[27] C. Castelluccia, P. Druschel, S. Hübner, S. Gorniak, D. Ikonomou, A. Pasic, B. Preneel, H. Tschofenig, and R. Tirtea, "Privacy, Accountability and Trust – Challenges and Opportunities," ENISA, Tech. rep., 2011.
[28] J. Camenisch, S. Fischer Hübner, and K. Rannenberg, *Privacy and Identity Management for Life*. Springer, 2011.
[29] C. Kahl, K. Böttcher, M. Tschersich, S. Heim, and K. Rannenberg, "How to Enhance Privacy and Identity Management for Mobile Communities: Approach and User Driven Concepts of the PICOS Project," in *Security and Privacy – Silver Linings in the Cloud*. Springer, 2010, pp. 277–288.