Preventing Information Inference in Access Control

Federica Paci
University of Southampton
f.m.paci@soton.ac.uk

Nicola Zannone
Eindhoven University of Technology
n.zannone@tue.nl

ABSTRACT

Technological innovations like social networks, personal devices and cloud computing, allow users to share and store online a huge amount of personal data. Sharing personal data online raises significant privacy concerns for users, who feel that they do not have full control over their data. A solution often proposed to alleviate users' privacy concerns is to let them specify access control policies that reflect their privacy constraints. However, existing approaches to access control often produce policies which either are too restrictive or allow the leakage of sensitive information. In this paper, we present a novel access control model that reduces the risk of information leakage. The model relies on a data model which encodes the domain knowledge along with the semantic relations between data. We illustrate how the access control model and the reasoning over the data model can be automatically translated in XACML. We evaluate and compare our model with existing access control models with respect to its effectiveness in preventing leakage of sensitive information and efficiency in authoring policies. The evaluation shows that the proposed model allows the definition of effective access control policies that mitigate the risks of inference of sensitive data while reducing users' effort in policy authoring compared to existing models.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Security, Theory

Keywords

Inference control; information leakage; semantic approach; XACML; comparison study

1. INTRODUCTION

Today individuals live in a digital world where everything they do happens online: they use their personal devices to check their email, read their favorite blogs, look for restaurants and jobs, read their friends' social network profiles, buy services and goods, tweet their locations, and more. However, everything individuals do online leaves a huge amount of information about them online. This makes it easy for companies and government agencies to collect this information as well as use analytic models to infer who a user is and what he does. For example, it is possible to discover the dietary preferences of an individual from the recipes she searches online; or to predict if she is pregnant based on her purchase habits. This information is considered highly sensitive, and an individual might want to disclose this information only to a restrict audience. Thus, the online proliferation of personal information raises significant privacy concerns for individuals.

Several studies have shown that individuals' privacy concerns are alleviated when they have a greater sense of control over the disclosure and subsequent use of their personal information [2, 13, 30]. A solution proposed to empower individuals with control over their personal information is to let them specify fine-grained policies that define which personal information they are willing to disclose and to whom it can be disclosed [4, 11, 16, 17, 39].

However, policy authoring has been proven to be a time consuming and error prone task [16, 20]. This is mainly due to the significant cognitive burden required by existing languages for policy authoring. In fact, in order to specify a policy, users need to have a deep and comprehensive knowledge of the application domain. To illustrate this issue, imagine a patient defining a policy to protect her Electronic Healthcare Record (EHR). Suppose that the patient wants to reveal that she is HIV-positive only to her treating doctor. To this end, she defines a policy explicitly restricting the access to this information, leaving other information in her EHR unrestricted. Based on this policy, no one except her treating doctor can read the HIV status in the patient's EHR. However, despite this policy, a nurse can read information concerning AIDS in the patient's EHR and, thus, infer that the patient is HIV-positive.

The main problem is that the disclosure of data, which a user may not consider sensitive and thus he leaves unrestricted, can allow the inference of sensitive information that the user wants to keep private [1, 3, 7, 18, 37]. A cause of this undesirable behavior is that users often ignore the semantic relation between data when specifying their policies [12, 34, 41]. Without such a knowledge, users can end up with policies that do not reflect their privacy inclinations: policies can be either too restrictive or expose them to the risk of disclosing their personal information to a much wider audience than intended. However, users often have only a limited knowledge of the application domain. For instance, an "average" patient unlikely has knowledge of all medical terms and the relations between them, which is necessary to specify accurate and effective access control policies to protect information in an EHR.

This observation motivates our research question: *How to facilitate users in the specification of access control policies that effectively protect them against inference of sensitive information?*

In this paper we make three contributions to the authoring and enforcement of access control policies able to capture users' privacy constraints. First, we propose an access control model that prevents inference of sensitive information caused by the semantic relations between data. In particular, the access control model is based on a semantic approach which leverages knowledge about the application domain for access decision making. We represent domain knowledge along with the semantic relations between data in a data model. Intuitively, the data model organizes data within an application domain in a hierarchical structure as well as it makes semantic inference relations between data explicit. Based on these relations, it is possible to reason on situations in which access control policies grant access to data which make it possible to infer sensitive information a user would like to keep private. To this end, we study how information can be inferred through semantic relations between data and define authorization propagation rules that prevent the inference of information to be protected. These propagation rules form an inference control mechanism. Such a mechanism mitigates the impact of mistakes that users can make in the specification of their policies due to inaccurate or partial knowledge they have about the application domain. The access control model does not require users to define the data model and, thus, to have a deep knowledge of the application domain. We show how existing domain specific ontologies can be leveraged for the representation of domain knowledge, relieving users of the burden of its definition.

Second, we demonstrate that the proposed access control model can be implemented using existing access control mechanisms. In particular, we show that the model can be automatically translated in XACML [33], the de facto standard for the specification and enforcement of access control policies. To this end, we provide an encoding of the access control model and reasoning over the data model in XACML.

Last, we compare the effectiveness and efficiency of existing access control models with respect to our model. Our findings show that access control models that do not consider semantic relations between data as the basis for authorization decision making do not fully protect users from inference of sensitive information and require significant efforts in writing policies. In contrast, our access control model prevents semantic inference of sensitive information while reducing users' effort in policy authoring in that it minimizes the number of policy statements a user has to specify.

The remainder of the paper is structured as follows. Section 2 introduces the basic notation used to formalize the proposed model. Section 3 reviews existing access control models and presents their formalization using the notation introduced in Section 2. Then, Section 4 presents the proposed access control model. Section 5 illustrates how the policies and the reasoning on the data model can be encoded in XACML. Section 6 compares the effectiveness and efficiency of various access control models. Finally, Section 7 reviews related work, and Section 8 concludes the paper providing directions for future work.

2. BASIC NOTATION

In this section we introduce the basic concepts underlying our approach. First, we present a simple notation for the representation of access control policies that suffices for the purpose of this work. Then, we introduce a formal definition of access control model.

An access control policy specifies the permissions that are granted or denied for each subject, i.e. which actions a subject can perform on an object. Our notation distinguishes *positive authorizations* and *negative authorizations* to allow easy management of exceptions in policy definition.

DEFINITION 1. *Let S be a set of subjects, A a set of actions, D a set of data elements and $R = \{+, -\}$ the set of rulings. An* access control policy *is a tuple $\langle s, a, d, r\rangle$ with $s \in S$, $a \in A$, $d \in D$, and $r \in R$. We refer to access control policies of the form $\langle s, a, d, +\rangle$ as* positive policies *and to policies of the form $\langle s, a, d, -\rangle$ as* negative policies.

The first three elements of a policy (i.e., subject, action and object) define the target of the policy. Intuitively, the target of a policy represents the applicability space of the policy, i.e. to which access requests the policy is applicable. The ruling defines the effect of the policy where $+$ indicates Permit and $-$ indicates Deny. A positive policy $\langle s, a, d, +\rangle$ states that subject s is allowed to execute action a on data element d. Similarly, a negative policy $\langle s, a, d, -\rangle$ states that subject s is not allowed to execute action a on data element d. Hereafter, we denote $\mathcal{P}$ the set of access control policies, $\mathcal{P}^+ \subseteq \mathcal{P}$ the set of positive policies in $\mathcal{P}$, and $\mathcal{P}^- \subseteq \mathcal{P}$ the set of negative policies in $\mathcal{P}$.

EXAMPLE 1. *An example of positive policy is*

$$\langle \mathit{doctor}, \mathit{read}, \mathit{medical\ record}, +\rangle$$

with subject "doctor"*, action* "read"*, object* "medical record" *and ruling "$+$". The policy should be read: "a doctor is allowed to read a medical record".*

The effectiveness of a policy depends on the access control model used for its evaluation. In fact, an access control model determines how policies are evaluated based on the data structures used to organize and reason on the policy elements in the target. We define formally an access control model as follows. Note that, as the focus of this work is on information inference, in the definition we only consider the data structure used to represent data elements. However, the definition can be easily extended to consider data structures for subjects (e.g., role hierarchies) and actions.

DEFINITION 2. *An* access control model *is a tuple $\langle \mathcal{Q}, \mathcal{P}, \mathcal{DS}, [\![\cdot]\!]_{\mathcal{DS}}\rangle$ where $\mathcal{Q} \subseteq S \times A \times D$ is a set of access requests, $\mathcal{P}$ is a set of policies, $\mathcal{DS}$ is a data structure encoding the relationships between data elements, and $[\![\cdot]\!]_{\mathcal{DS}} : \mathcal{Q} \times \wp(\mathcal{P}) \rightarrow \{$Permit, Deny, NotApplicable$\}$ is an evaluation function mapping an access request to an access decision based on $\mathcal{P}$ and $\mathcal{DS}$.*

An evaluation function should determine the applicability of policies for a given access request and, based on the applicable policies, map the request to an access decision. We use the predicate $match(q, \mathcal{P})$ to denote that there is a policy $P \in \mathcal{P}$ that directly matches a request $q \in \mathcal{Q}$, i.e. the elements in the request match the elements in the policy. The predicate $match$ can be defined in the straightforward way over the target of a policy. In this work, we are mainly interested to study the behavior of access control models with respect to data elements. Thus, abusing notation, we will write $[\![d, \mathcal{P}]\!]_{\mathcal{DS}}$ to represent the evaluation of an access request for data element d against a set of policies $\mathcal{P}$, and $match(d, \mathcal{P})$ to represent there exists a policy $P \in \mathcal{P}$ that matches a request for d.

The combined use of both positive and negative authorizations can lead to conflicts when conflicting policies are applicable for

Reference Model	Data Structure	Evaluation Function	Existing Models
$\mathcal{AC}_{\mathcal{NR}} = \langle \mathcal{Q}, \mathcal{P}, \mathcal{NR}, \llbracket\cdot\rrbracket_{\mathcal{NR}} \rangle$	$\mathcal{NR}$: No relations	$\llbracket d, \mathcal{P} \rrbracket_{\mathcal{NR}} = \begin{cases} \text{Deny} & \text{if } match(d, \mathcal{P}^-) \\ \text{Permit} & \text{if } match(d, \mathcal{P}^+) \wedge \llbracket d, \mathcal{P} \rrbracket_{\mathcal{NR}} \neq \text{Deny} \\ \text{NotApplicable} & \text{otherwise} \end{cases}$	System R [6]
$\mathcal{AC}_{\mathcal{DH}_1} = \langle \mathcal{Q}, \mathcal{P}, \mathcal{DH}, \llbracket\cdot\rrbracket_{\mathcal{DH}_1} \rangle$	$\mathcal{DH}$: Data Hierarchy	$\llbracket d, \mathcal{P} \rrbracket_{\mathcal{DH}_1} = \begin{cases} \text{Deny} & \text{if } \exists d_i \in D \text{ s.t. } d_i \in d^{\uparrow} \wedge match(d, \mathcal{P}^-) \\ \text{Permit} & \text{if } \big(\exists d_i \in D \text{ s.t. } d_i \in d^{\uparrow} \wedge match(d, \mathcal{P}^+)\big) \\ & \quad \wedge \llbracket d, \mathcal{P} \rrbracket_{\mathcal{DH}_1} \neq \text{Deny} \\ \text{NotApplicable} & \text{otherwise} \end{cases}$	FAF [23] WDAP [36] PBAC [11] Lee et al. [27] Masoumzadeh et al. [28]
$\mathcal{AC}_{\mathcal{DH}_2} = \langle \mathcal{Q}, \mathcal{P}, \mathcal{DH}, \llbracket\cdot\rrbracket_{\mathcal{DH}_2} \rangle$	$\mathcal{DH}$: Data Hierarchy	$\llbracket d, \mathcal{P} \rrbracket_{\mathcal{DH}_2} = \begin{cases} \text{Deny} & \text{if } \exists d_i \in D \text{ s.t. } d_i \in d^{\downarrow} \wedge match(d, \mathcal{P}^-) \\ \text{Permit} & \text{if } \big(\exists d_i \in D \text{ s.t. } d_i \in d^{\uparrow} \wedge match(d, \mathcal{P}^+)\big) \\ & \quad \wedge \llbracket d, \mathcal{P} \rrbracket_{\mathcal{DH}_2} \neq \text{Deny} \\ \text{NotApplicable} & \text{otherwise} \end{cases}$	Rabitti et al. [35]
$\mathcal{AC}_{\mathcal{DH}_3} = \langle \mathcal{Q}, \mathcal{P}, \mathcal{DH}, \llbracket\cdot\rrbracket_{\mathcal{DH}_3} \rangle$	$\mathcal{DH}$: Data Hierarchy	$\llbracket d, \mathcal{P} \rrbracket_{\mathcal{DH}_3} = \begin{cases} \text{Deny} & \text{if } \exists d_i \in D \text{ s.t. } d_i \in d^{\updownarrow} \wedge match(d, \mathcal{P}^-) \\ \text{Permit} & \text{if } \big(\exists d_i \in D \text{ s.t. } d_i \in d^{\uparrow} \wedge match(d, \mathcal{P}^+)\big) \\ & \quad \wedge \llbracket d, \mathcal{P} \rrbracket_{\mathcal{DH}_3} \neq \text{Deny} \\ \text{NotApplicable} & \text{otherwise} \end{cases}$	EPAL [4] DPAL [5]
$\mathcal{AC}_{\mathcal{DM}} = \langle \mathcal{Q}, \mathcal{P}, \mathcal{DM}, \llbracket\cdot\rrbracket_{\mathcal{DM}} \rangle$	$\mathcal{DM}$: Data Model	$\llbracket d, \mathcal{P} \rrbracket_{\mathcal{DM}} = \begin{cases} \text{Deny} & \text{if } \big(\exists d_i \in D \text{ s.t. } d_i \in d^{\downarrow} \wedge match(d_i, \mathcal{P}^-)\big) \vee \\ & \quad \big(\exists d_i \in D \text{ s.t. } d_i \in d^{\infty} \wedge match(d_i, \mathcal{P}^-)\big) \\ \text{Permit} & \text{if } \exists d_i \in D \text{ s.t. } \big(d_i \in d^{\uparrow} \wedge match(d_i, \mathcal{P}^+)\big) \\ & \quad \wedge \llbracket d, \mathcal{P} \rrbracket_{\mathcal{DM}} \neq \text{Deny} \\ \text{NotApplicable} & \text{otherwise} \end{cases}$	this work

Table 1: Overview of access control models

the same data element. Several conflict resolution strategies have been proposed in the literature [4, 23, 29]. As our goal is to prevent leakages of sensitive information, we assume that an evaluation function combines conflicting (applicable) policies using a deny-overrides strategy, where negative authorizations override positive authorizations.

3. REVIEW OF ACCESS CONTROL MODELS

Several access control models have been proposed in the literature to regulate the access to sensitive information. Access control models use an evaluation function to determine the policies that apply to a given access request and, based on the applicable policies, make an authorization decision. The definition of such an evaluation function depends on the data structure used by the access control model to reason on the elements in the target of a policy. In the remainder of this section, we review a number of existing access control models, focusing on the data element as the main decision criterion. An overview of the considered models is presented in Table 1. Note that, for the sake of comparison, we limit our study to access control models that explicitly support both positive and negative authorizations.

Early access control models like the one presented in [6] do not consider relationships between data elements for decision making. In these models, a policy is applied to an access request for a data element only if the policy has been explicitly specified for that data element. Hereafter, $\llbracket\cdot\rrbracket_{\mathcal{NR}}$ denotes a policy evaluation function that does not use relations between data elements to make access decision, and $\mathcal{AC}_{\mathcal{NR}}$ access control models that use such a function (a formal definition of $\llbracket\cdot\rrbracket_{\mathcal{NR}}$ and $\mathcal{AC}_{\mathcal{NR}}$ is given in Table 1).

To effectively reduce the number of permission assignments (and thus reducing the cost of policy administration), several access control models [4, 11, 23, 38, 35] represent and reason on data elements in a policy using hierarchical structures.

DEFINITION 3. *A* data hierarchy $\mathcal{DH}$ *is a pair* $\langle D, \uparrow \rangle$ *where*

- *D is a set of data elements;*
- *$\uparrow \subseteq D \times D$ a partial order on D.*

A data hierarchy $\mathcal{DH}$ organizes data elements in direct acyclic graph (DAG). Hierarchy relations represent a specialization relationship between data elements, i.e. $(d', d) \in \uparrow$ denotes that d' is a specialization of d. Based on hierarchy relations we define the descendants and ancestors of a node d. A data element d' is a *descendant* of a data element d if d' is either a child of d (i.e., $(d', d) \in \uparrow$) or the child of some descendants of d. A data element d' is an *ancestor* of a data element d if d' is either a parent of d (i.e., $(d, d') \in \uparrow$) or the parent of some ancestors of d. The set of descendants of a data element d, including the data element itself, is denoted $d^{\downarrow}$. The set of ancestors of a data element d, including the data element itself, is denoted $d^{\uparrow}$. $d^{\updownarrow}$ denotes the set of ancestors or descendants of d, i.e. $d^{\updownarrow} = d^{\downarrow} \cup d^{\uparrow}$.

Different propagation rules over data hierarchies have been proposed for policy evaluation. Some access control models (e.g., [23, 27, 28, 36]) assume that both positive and negative authorizations are propagated down the data hierarchy.[1] Intuitively, positive and negative authorizations propagate to the descendants of the data element specified in a policy. Byun and Li [11] annotate data elements with intended purpose labels (representing the allowed and prohibited intended usage of data), and use these labels to control access to data elements. To determine the effective intended purpose of data elements, they present a purpose inference mechanism that propagates the intended purpose of a data element to its child elements, thus propagating authorizations down the data hierarchy.[2] We use $\llbracket\cdot\rrbracket_{\mathcal{DH}_1}$ to denote the evaluation function using these propagation rules and $\mathcal{AC}_{\mathcal{DH}_1}$ the corresponding access control models (see Table 1 for their formalization).

Other access control models (e.g., [35]) use different propagation rules for negative authorizations. In particular, these models propagate negative authorizations up the data hierarchy, i.e. negative authorizations are propagate to the ancestors of the data ele-

[1]Note that a number of propagation policies are presented in [23], namely *no propagation*, *no overriding*, *most specific overrides* and *path overrides*. In this work, we consider the *no overrides* policy in which all the authorizations of a node are propagated to its child nodes, regardless of the presence of other contradicting authorizations. However, it is worth noting that the other policies also propagate authorizations down the data hierarchy.

[2]The work in [11] also uses propagations rules which support the inheritance of negative authorizations up the hierarchy like in [4, 5]. However, these rules are limited to the reasoning over the purpose hierarchy and are not used to reason over the data hierarchy.

ments for which a negative policy is defined.[3] In the remainder, we denote $[\![\cdot]\!]_{\mathcal{DH}_2}$ the evaluation function based on these propagation rules and $\mathcal{AC}_{\mathcal{DH}_2}$ the corresponding access control models. Their formalization is given in Table 1.

Some access control models like EPAL [4] and DPAL [5], combine the propagation rules underlying $\mathcal{AC}_{\mathcal{DH}_1}$ and $\mathcal{AC}_{\mathcal{DH}_2}$. These models propagate positive authorizations down the data hierarchy. On the other hand, negative authorizations are inherited both up and down the data hierarchies. Intuitively, negative authorizations are propagated to the ancestors and descendants of the data element for which a negative policy is defined. In the remainder, we denote $[\![\cdot]\!]_{\mathcal{DH}_3}$ the evaluation function based on these propagation rules and $\mathcal{AC}_{\mathcal{DH}_3}$ the corresponding access control models. Their formalization is given in Table 1.

4. PREVENTING INFORMATION INFERENCE

The access control models discussed in the previous section are not able to protect end-users from situations in which other users can infer sensitive information from information to which they have legitimate access. The main reason is that they do not account for the semantic relations between data elements when making authorization decisions. For example, it is possible to infer whether a patient is HIV-positive by knowing that the patient has AIDS or the T-helper cell count[4]. Thus, an access control model that does not rely on the semantic relation between data elements, will inevitably lead to information leakage.

To address these issues, we propose a semantic approach which enables to reason on the information that can be inferred from the disclosure of a data element. The approach is based on a data model that augments a data hierarchy with inference relations between data elements.

DEFINITION 4. *A* data model $\mathcal{DM}$ *is a tuple* $\langle D, \uparrow, \rightarrow \rangle$ *where*

- *D is a set of data elements;*
- *$\uparrow \subseteq D \times D$ is a partial order on D;*
- *$\rightarrow \subseteq D \times D$ represents an inference relation on D.*

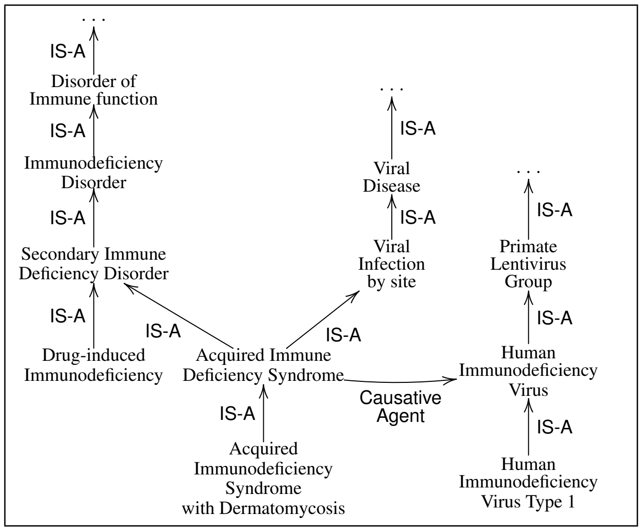
Intuitively, a data model encodes the domain knowledge, making the semantic relation between data elements within the application domain explicit. As discussed in Section 3, hierarchy relations represent the specialization relation between data elements. Inference relations $(d', d) \in \rightarrow$ indicate that by knowing a data element d', a user knows data element d. We denote $\rightarrow^R$ the reflexive closure of $\rightarrow$. We say that a data element d' is *reachable* from a data element d, denoted $(d, d') \in \rightsquigarrow$, if and only if $\exists d_i \in d^\uparrow$ s.t. $(d_i, d') \in \rightarrow^R$. We denote $\rightsquigarrow^*$ the transitive closure of $\rightsquigarrow$. d^∞ denotes the set of data elements that are reachable from a data element d, i.e. $d^\infty = \bigcup_{(d,d_i)\in\rightsquigarrow^*} d_i^\uparrow$.

It is worth noting that our approach does not require policy authors to define the data model and thus to have a deep knowledge of the application domain. Domain knowledge is often represented using an ontology, and several ontologies are currently available for a large range of application domains, e.g. FOAF[5] for social networks, GoodRelations [19] and CContology [24] for e-commerce,

[3]The access control model in [35] uses special actions, e.g. *Read Definition*, for which positive authorizations are propagated up the data hierarchy and negative authorizations are propagated down the hierarchy. We do not consider these actions in our model.

[4]The HIV virus attacks T-helper cells, destroying them.

[5]http://www.foaf-project.org


...
IS-A
Disorder of Immune function
IS-A
Immunodeficiency Disorder
IS-A
Secondary Immune Deficiency Disorder
IS-A
Drug-induced Immunodeficiency
IS-A
Acquired Immune Deficiency Syndrome
IS-A
Acquired Immunodeficiency Syndrome with Dermatomycosis
...
IS-A
Viral Disease
IS-A
Viral Infection by site
IS-A
Causative Agent
...
IS-A
Primate Lentivirus Group
IS-A
Human Immunodeficiency Virus
IS-A
Human Immunodeficiency Virus Type 1

(a) Fragment of SNOMED-CT

Data Model	**SNOMED-CT**
Nodes	Concepts
Hierarchy relations ($\uparrow$)	IS-A relationship
Inference relations ($\rightarrow$)	Attribute relationships

(b) Mapping of the data model to SNOMED-CT

Figure 1: Data model instantiation in the healthcare domain.

and many others. These ontologies can be used as a representation of the data model. Next, we show an exemplification of how the data model can be implemented through an ontology within the healthcare domain.

EXAMPLE 2. *Let us illustrate how the proposed data model can be instantiated in the healthcare domain using SNOMED Clinical Terms (SNOMED-CT) [21], one of the most complete and used medical ontologies. SNOMED-CT provides a comprehensive and structured collection of medical terms often used in electronic health records along with relations between them. SNOMED-CT includes terms related to clinical findings, symptoms, diagnoses, procedures, body structures, organisms and other etiologies, substances, pharmaceuticals, devices and specimen. Fig. 1a presents a fragment of SNOMED-CT, and Fig. 1b the mapping of the data model to SNOMED-CT. In SNOMED-CT a* concept *is a clinical meaning identified by a unique identifier. The meaning of concepts is defined in terms of the relationships with other concepts. SNOMED-CT uses the* IS-A *relationship to represent hierarchical associations between entity types (i.e., generalization/specialization relationship). In particular, this relationship organizes concepts into DAGs where concepts can have multiple parent nodes. For instance, the term* Acquired Immune Deficiency Syndrome *(AIDS) is child of both terms* Secondary Immune Deficiency Disorder *and* Viral Infection by site. *SNOMED-CT also provides a number of attribute relationships which define interrelationships between concepts. In particular, an attribute relationship is an association between two concepts describing an intrinsic property of the concepts. Attribute relationships can be used to infer additional information about a data element. For instance, SNOMED-CT relates the term* Acquired Immune Deficiency Syndrome *to the term* Human Immunodeficiency Virus *(HIV) using attribute relationship* Causative Agent*; knowing that a patient has AIDS reveals that she is HIV-positive.*

Given a data model $\mathcal{DM}$, we are interested in an access control model $\langle \mathcal{Q}, \mathcal{P}, \mathcal{DM}, [\![\cdot]\!]_{\mathcal{DM}} \rangle$ which uses $\mathcal{DM}$ to make authorization decisions. To this end, we have to define an evaluation function $[\![\cdot]\!]_{\mathcal{DM}}$ which allows propagation of positive and negative authorizations along with hierarchical and inference relations in the data model. For the propagation through data hierarchies, we adopt the approach proposed in [4, 5]. As shown in Section 6, other propagation rules over data hierarchies do not capture user intention properly and/or require additional effort for policy authoring.

Inference relationships only propagate negative authorizations (in the opposite direction of the inference). The reason is that by explicitly denying the access to a data element a policy author intends to restrict the access to such a data element. Therefore, the access to all data elements that allow its inference should be restricted. On the other hand, allowing the access to a data element does not imply that the access to data elements from which it can be inferred should be allowed. We can observe that the set of data elements that can be inferred by a data element is the set of data elements reachable from that data element. Indeed, by definition, a data element is reachable from another data element if the former can be inferred by the latter. Moreover, reachability is defined over bottom-up propagation of negative authorizations through data hierarchies. This is motivated by the fact that knowledge of a data element makes it possible to infer information on the ancestors of that data element. Conversely, knowledge of a data element does not imply a leakage of information about the descendants of that data element. For instance, knowing that a patient has Disorder of Immune function does not allow a nurse to infer that the patient has Acquired Immune Deficiency Syndrome. Indeed the patient might be affected by Drug-induced Immunodeficiency or other diseases (not represented in Fig. 1a).

The following definition formally defines the evaluation function $[\![\cdot]\!]_{\mathcal{DM}}$.

DEFINITION 5. *Let $\mathcal{DM} = \langle D, \uparrow, \rightarrow \rangle$ be a data model and $\langle \mathcal{Q}, \mathcal{P}, \mathcal{DM}, [\![\cdot]\!]_{\mathcal{DM}} \rangle$ an access control model using $\mathcal{DM}$ for decision making. Given a request for data element $d \in D$, the evaluation function $[\![\cdot]\!]_{\mathcal{DM}}$ is*

$$[\![d, \mathcal{P}]\!]_{\mathcal{DM}} = \begin{cases} \text{Deny} & \textit{if } match(d, \mathcal{P}^-) \vee \\ & \quad \big(\exists d_i \in D \textit{ s.t. } d_i \in d^{\updownarrow} \wedge \\ & \quad match(d_i, \mathcal{P}^-)\big) \vee \\ & \quad \big(\exists d_i \in D \textit{ s.t. } d_i \in d^{\infty} \wedge \\ & \quad match(d_i, \mathcal{P}^-)\big) \\ \text{Permit} & \textit{if } \big[match(d, \mathcal{P}^+) \vee \\ & \quad \exists d_i \in D \textit{ s.t. } \big(d_i \in d^{\uparrow} \wedge \\ & \quad match(d_i, \mathcal{P}^+)\big)\big] \wedge \\ & \quad [\![d, \mathcal{P}]\!]_{\mathcal{DM}} \neq \text{Deny} \\ \text{NotApplicable} & \textit{otherwise} \end{cases}$$

The definition of the evaluation function $[\![\cdot]\!]_{\mathcal{DM}}$ explicitly encodes the propagation rules discussed above. It is worth noting that such a definition is redundant. For instance, by definition $d^{\downarrow}$, $d^{\updownarrow}$ and d^{∞} include data element d. Moreover, $d^{\updownarrow}$ and d^{∞} are not disjoint, i.e. $d^{\updownarrow} \cap d^{\infty} = d^{\uparrow}$. Based on these observations, the evaluation function $[\![\cdot]\!]_{\mathcal{DM}}$ can be rewritten as shown at the end of Table 1.

EXAMPLE 3. *Consider an Electronic Healthcare Record (EHR) whose fields and entries are defined over SNOMED-CT. Let us suppose that the record belongs to a patient named Alice who is affected by AIDS. Alice is highly concerned about revealing she has contracted a virus in the primate lentivirus group. Thus, she defines a negative authorization stating that access to her status concerning* Primate Lentivirus Group *in her EHR is denied to any nurse. At the same time, Alice allows nurses working in the hospital where she is hospitalized to access information about* Immunodeficiency Disorder *in order to receive proper treatment. These authorizations are represented by the following policies*

$$P_1 = \langle nurse, read, \text{Primate Lentivirus Group}, - \rangle$$
$$P_2 = \langle nurse, read, \text{Immunodeficiency Disorder}, + \rangle$$

If only data hierarchies are used to evaluate access requests, a nurse is allowed to know that the patient has AIDS. Indeed, the positive authorization defined by policy P_2 is propagated to the descendants of Immunodeficiency Disorder *and, thus, to* Acquired Immune Deficiency Syndrome. *This, however, allows the nurse to infer that Alice has contracted a virus in the primate lentivirus group, violating policy P_1. Reasoning over inference relationships makes it possible to propagate the negative authorization associated with* Primate Lentivirus Group *to* Acquired Immune Deficiency Syndrome. *As authorizations are combined using deny-overrides, the access to* Acquired Immune Deficiency Syndrome *is denied and, thus, the actual permissions on the data element comply with Alice's privacy constraints.*

We remark that the aim of this work is the design of an access control model that prevents inference of sensitive information due to explicit inference relations between data elements. In this setting, the absence of an inference relation does not necessarily indicate that a data element cannot be inferred, for instance using statistical methods.

5. POLICY ENCODING AND ENFORCING IN XACML

In this section we demonstrate how the proposed access control model can be implemented using existing access control mechanisms. In particular, we present an encoding of the access control model in eXtensible Access Control Markup Language (XACML) [33], the de facto standard for the specification and enforcement of access control policies. The advantage of encoding the access control model in XACML is that there exist several XACML policy engines, many of which are free to use. These engines usually support basic functionalities needed for policy evaluation like the matching of access requests against policies.

Next, we first present the encoding of authorization policies defined using the notation presented in Section 2; then, we present how these policies are combined to support the reasoning on the data model presented in Section 4.

5.1 Encoding Policies in XACML

We show here how authorization policies written in the notation introduced in Section 2 can be easily transformed into enforceable policies written in XACML. Here we only discuss the elements needed to encode our access control model and refer to [33] for the complete XACML policy language specification.

Positive and negative authorizations are represented using the `<Rule>` element whose `Effect` attribute is set respectively to `Permit` and `Deny`. Intuitively, the `Effect` attribute encodes the ruling of a policy (Definition 1). The subject s and action a are represented as attributes and, thus, mapped to `<Match>` elements in `<Target>` of the `<Rule>` element. The encoding of data element d requires considering the propagation rules; its encoding is presented in the next section.

Rules, however, are not intended to be evaluated in isolation by the policy decision point (PDP) in XACML. The basic unit of policy used by the PDP for evaluation is the `<Policy>` element [33,

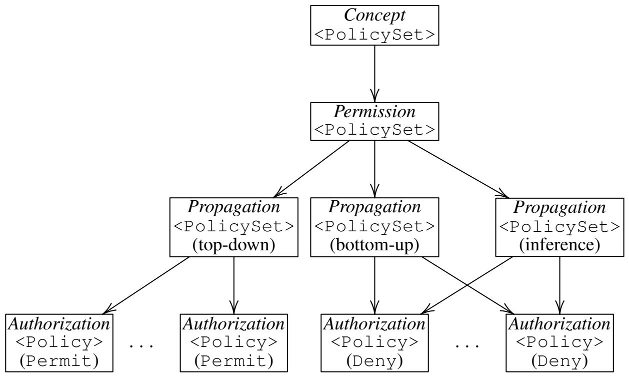


Figure 2: Policy Structure

Section 2.2]. To this end, we wrap each `<Rule>` element into a `<Policy>` element (with an empty `<Target>`). In the next section, we discuss how these policies are combined.

5.2 Encoding the Data Model in XACML

A natural choice to represent data hierarchies in XACML would be to adopt the XACML Hierarchical Resource Profile [32], which illustrates how to specify XACML policies for resources that are structured as hierarchies. However, this profile does not distinguish the propagation of positive and negative authorizations, which is necessary to implement the propagation strategy over the data model. To this end, we represent the data model in terms of the policy structure along the lines suggested in the XACML RBAC Profile [31] to deal with role hierarchies. This profile distinguished between *Permission* `<PolicySet>`, which contains the actual permissions associated with a role, and *Role* `<PolicySet>`, which associates the role with the corresponding permissions. A Permission `<PolicySet>` can contain references to the Permission `<PolicySet>` elements associated with other roles, thus enabling permission propagation through role hierarchies.

We use a similar approach to enable the permission propagation over the data model. Every data element in the data model is associated with a XACML policy encoding all the policies specified for the data element. These XACML policies consist of four parts (Fig. 2): a *Concept* `<PolicySet>` element, a *Permission* `<PolicySet>` element, *Propagation* `<PolicySet>` elements, and *Authorization* `<Policy>` elements. The Concept `<PolicySet>` specifies the data element to which the policy applies, while the Permission `<PolicySet>` is used to manage the actual authorizations over such a data element. The data element is specified as an attribute (i.e., a `<Match>` element) in the `<Target>` element of a Concept `<PolicySet>`; the Concept `<PolicySet>` element references a single Permission `<PolicySet>`, which in turn contains references to Propagation `<PolicySet>` elements. In turn, Propagation `<PolicySet>` elements reference Authorization `<Policy>` elements which encode the actual authorizations over the data element as described in Section 5.1. The extra layer given by the Propagation `<PolicySet>` elements is needed to prevent loops in the policy structure due to up and down inheritance through data hierarchies and thus to properly implement the evaluation function $[\![\cdot]\!]_{\mathcal{DM}}$.

The XACML policy associated with a data element contains three Propagation `<PolicySet>` elements, one for top-down inheritance over data hierarchies, one for bottom-up inheritance over data hierarchies and one for inference (defined in terms of reachability). The top-down Propagation `<PolicySet>` contains references to the positive Authorization `<Policy>` elements, i.e. to `<Policy>` elements that only contain a `Permit` rule.[6] In addition, it contains a policy reference to the top-down Propagation `<PolicySet>` associated with the parent node(s) in the data model. This way, the data element inherits the (positive) policies of the parent node(s). The bottom-up and inference Propagation `<PolicySet>` elements are only linked to negative Authorization `<Policy>` elements, i.e. to `<Policy>` elements that only contain a `Deny` rule. This construction guarantees that only the propagation of negative authorizations can occur up a hierarchy or through inference relations. To inherit the negative authorizations of the child node(s), the bottom-up Propagation `<PolicySet>` also contains a policy reference to the bottom-up Propagation `<PolicySet>` of the child node(s) in the data model. Inference relations and top-down propagation of negative authorizations are encoded in a similar way. Let x be the node for which the policy is defined. For each inference relation $(x, y) \in \rightarrow$, a policy reference to the inference Propagation `<PolicySet>` associated with the node y is added to the inference Propagation `<PolicySet>` associated with x. The inference Propagation `<PolicySet>` also contains a reference to the inference Propagation `<PolicySet>` of the parent node(s). This combination of inference and hierarchical relations captures the notion of reachability, thus allowing the inheritance of the (negative) authorizations associated with the data elements that can be inferred from the disclosure of the data element.

To enable propagation through data hierarchies and inference relations, the `<Target>` element of Permission `<PolicySet>` elements should not constraint the data element to which the policy element applies. Consequently, Permission `<PolicySet>`, Propagation `<PolicySet>` and Authorization `<Policy>` elements cannot be root policies, i.e. they cannot be directly evaluated by the PDP. Policy elements are combined using the `deny-overrides` combining algorithm. Intuitively, if one of these policies evaluates `Deny`, the access to the data element is denied.

Encoding the evaluation function within the policy structure makes it possible to build a simple and automated procedure for the construction of XACML policies implementing the evaluation function $[\![\cdot]\!]_{\mathcal{DM}}$. In addition, this approach provides the flexibility necessary to support the adoption of different strategies for policy evaluation. For instance, it allows the implementation of an evaluation strategy based on the specificity level of policies [23, 29]. Intuitively, this strategy gives higher priority to policies that are more specific. Specificity can be implemented using the `first-applicable` combining algorithm. In particular, the policies associated with a data element and with the data elements that can be inferred from it can be combined using `deny-overrides`. The resulting policy can be combined with the policies of the parent node(s) using `first-applicable`. This ensures that the policy of parent node(s) is evaluated only if the policies specific for the data element are not applicable.

It is worth noting that, although the proposed encoding requires the definition of a XACML policy for each data element in the application domain, only the Concept `<PolicySet>` concerning the requested data element is applicable. Thus, only the applicable Concept `<PolicySet>` and the policy elements reachable from it are evaluated, making policy evaluation independent from the size of the application domain in terms of data elements.

[6]Remark that in our construction Authorization `<Policy>` elements contain a single rule.

6. EVALUATION AND COMPARISON

In this section we analyze the effectiveness and efficiency of the access control models in Table 1 with respect to $\mathcal{AC}_{\mathcal{DM}}$. In particular, we study the impact of using inference relations on the specification and evaluation of access control policies. *Effectiveness* characterizes the completeness and accuracy of access control policies, while *efficiency* characterizes the effort (in relation to the completeness and accuracy) needed to specify the policies [22].

Evaluation Framework.

To measure the effectiveness and efficiency of an access control model, we introduce an evaluation framework consisting of a number of metrics. These metrics are defined in terms of *explicitly protected data* and *intended protected data*. To formally define the metrics, we introduce the following notation.

Let $\mathcal{DM} = \langle D, \uparrow, \rightarrow \rangle$ be a data model and $\mathcal{P}$ a set of policies. The set of data elements explicitly protected by $\mathcal{P}$ is the set of data elements for which a policy is defined. Specifically, we have $D^{\mathcal{P}^+}_{Exp} = \{d : match(d, \mathcal{P}^+)\}$ and $D^{\mathcal{P}^-}_{Exp} = \{d : match(d, \mathcal{P}^-)\}$. The set of intended protected data is the set of data elements whose access needs to be regulated in order to capture a user's privacy constraints according to the domain knowledge encoded in $\mathcal{DM}$. Specifically, we have $D^{\mathcal{P}^-}_{Int} = \{d_i : d_i \in d^{\downarrow} \wedge d \in D^{\mathcal{P}^-}_{Exp}\} \cup \{d_i : d_i \in d^{\infty} \wedge d \in D^{\mathcal{P}^-}_{Exp}\}$ and $D^{\mathcal{P}^+}_{Int} = \{d_i : d_i \in d^{\uparrow} \wedge d \in D^{\mathcal{P}^+}_{Exp}\} \setminus D^{\mathcal{P}^-}_{Int}$. In other words, the set of intended protected data correspond to the evaluation function $[\![\cdot]\!]_{\mathcal{DM}}$.

Let $\mathcal{AC} = \langle \mathcal{Q}, \mathcal{P}, \mathcal{DS}, [\![\cdot]\!]_{\mathcal{DS}} \rangle$ be an access control model. The *accuracy* of $\mathcal{AC}$ is assessed using the following metrics:

(M_1) Number of cases where a user intends to deny the access to a data element, while access is not denied by the access control model.

$$M_1 = \left|D^{\mathcal{P}^-}_{Int} \setminus \{d \in D : [\![d, \mathcal{P}]\!]_{\mathcal{DS}} = \mathsf{Deny}\}\right|$$

(M_2) Number of cases where a user intends to permit the access to a data element, while access is not granted by the access control model.

$$M_2 = \left|D^{\mathcal{P}^+}_{Int} \setminus \{d \in D : [\![d, \mathcal{P}]\!]_{\mathcal{DS}} = \mathsf{Permit}\}\right|$$

(M_3) Number of cases where data elements could be leaked.

$$M_3 = \left|D^{\mathcal{P}^-}_{Int} \cap \{d \in D : [\![d, \mathcal{P}]\!]_{\mathcal{DS}} = \mathsf{Permit}\}\right|$$

(M_4) Number of cases where the availability of data elements is not guaranteed.

$$M_4 = \left|D^{\mathcal{P}^+}_{Int} \cap \{d \in D : [\![d, \mathcal{P}]\!]_{\mathcal{DS}} = \mathsf{Deny}\}\right|$$

These metrics aims to verify to what extent a policy evaluated with respect to a given access control model is able to capture a user's intention. In particular, M_1 checks whether a policy is less restrictive than what the user wants, while M_2 checks whether a policy is more restrictive than what the user wants. M_3 refines M_1 by checking whether access that should be denied is instead permitted. Similarly, M_4 refines M_2 by checking whether access that should be permitted is instead denied. In order to achieve accuracy, the objective of an access control model is to minimize these metrics.

The *completeness* of $\mathcal{AC}$ is assessed by measuring the coverage of $\mathcal{P}$. Coverage is evaluated using the following metric:

(M_5) Fraction of data elements that are correctly protected by $\mathcal{P}$ over the set of intended protected data.

$$M_5 = \frac{\left|\left(D^{\mathcal{P}^+}_{Int} \cap \{d \in D : [\![d, \mathcal{P}]\!]_{\mathcal{DS}} = \mathsf{Permit}\}\right) \cup \left(D^{\mathcal{P}^-}_{Int} \cap \{d \in D : [\![d, \mathcal{P}]\!]_{\mathcal{DS}} = \mathsf{Deny}\}\right)\right|}{\left|D^{\mathcal{P}^+}_{Int} \cup D^{\mathcal{P}^-}_{Int}\right|}$$

Intuitively, coverage determines to what extent a policy evaluated in a given access control model is able to capture the set of intended protected data. In order to achieve completeness, the objective of an access control model is to maximize this metric.

The *efficiency* of $\mathcal{AC}$ is measured by the following metric:

(M_6) Number of policies to be specified by a user in order to capture her intention.

$$M_6 = \min \left\{ \mid D^{\mathcal{P}^+}_{Ext} \mid + \mid D^{\mathcal{P}^-}_{Ext} \mid \text{s.t. } M_5 = 1 \right\}$$

The number of policies explicitly specified by a user quantifies the effort that the user has to spend to cover the set of intended protected data. Higher is the number of policies, higher is the complexity of specifying policies for the user. Therefore, in order to achieve efficiency, the objective of an access control model is to minimize this metric.

Evaluation Settings.

We have compared a number of existing access control models with $\mathcal{AC}_{\mathcal{DM}}$. In particular, we have analyzed the impact of the data model on the specification and evaluation of access control policies with respect to other data structures and propagation rules (Table 1). Recall that $\mathcal{AC}_{\mathcal{NR}}$ is representative of access control models that do not use relations between data elements, while $\mathcal{AC}_{\mathcal{DH}_1}$, $\mathcal{AC}_{\mathcal{DH}_2}$ and $\mathcal{AC}_{\mathcal{DH}_3}$ are representative of models which use data hierarchies. As discussed in Section 3, these models differ in the way negative authorizations are propagated.

The access control models in Table 1 and the evaluation framework have been implemented in Datalog. In particular, the Datalog program encodes the evaluation function (i.e., propagation rules and combining algorithm) for each access control model as well as the metrics M_1 to M_6.

To evaluate and compare the completeness, accuracy and efficiency of the access control models, we generated three sets of policies $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$ protecting access to an EHR specified over a vocabulary consisting of 100 terms (taken from the SNOMED-CT ontology). Each set of policies $\mathcal{P}_i$ ($i = 1, \ldots, 3$) defines the set of data elements to which access is explicitly permitted (i.e., $D^{\mathcal{P}^+_i}_{Ext}$) and denied (i.e., $D^{\mathcal{P}^-_i}_{Ext}$) by a user. From these sets, we computed the sets of intended protected data $D^{\mathcal{P}^+_i}_{Int}$ and $D^{\mathcal{P}^-_i}_{Int}$. Sets $D^{\mathcal{P}^+_i}_{Int}$ and $D^{\mathcal{P}^-_i}_{Int}$ have been used as a reference model for the evaluation.

To study the efficiency of access control models and their ability to capture user intention, for every set of policies $\mathcal{P}_i$ ($i = 1, \ldots, 3$) and access control model $\mathcal{AC}_j$ ($j = \mathcal{NR}, \mathcal{DH}_1, \mathcal{DH}_2, \mathcal{DH}_3, \mathcal{DM}$), we specified a policy set $\mathcal{P}^j_i$ that captures $D^{\mathcal{P}^+_i}_{Int}$ and $D^{\mathcal{P}^-_i}_{Int}$ using the minimal number of policy statements with respect to $\mathcal{AC}_j$. When an access control model is not able to fully cover both $D^{\mathcal{P}^+_i}_{Int}$ and $D^{\mathcal{P}^-_i}_{Int}$, we defined the minimal policy set that fully cover $D^{\mathcal{P}^-_i}_{Int}$ while maximizing the coverage of $D^{\mathcal{P}^+_i}_{Int}$. This represents the minimal policy set that prevent the risk of data leakage with respect to the access control model.

The three policy sets were passed as an input to the Datalog program which evaluated them with respect to the access control models in Table 1 and returned the metrics M_1 to M_6 for each access control model.

	$\mathcal{AC}_{\mathcal{NR}}$						$\mathcal{AC}_{\mathcal{DH}_1}$						$\mathcal{AC}_{\mathcal{DH}_2}$						$\mathcal{AC}_{\mathcal{DH}_3}$						$\mathcal{AC}_{\mathcal{DM}}$					
$\mathcal{P}_i$	M_1	M_2	M_3	M_4	M_5	M_6	M_1	M_2	M_3	M_4	M_5	M_6	M_1	M_2	M_3	M_4	M_5	M_6	M_1	M_2	M_3	M_4	M_5	M_6	M_1	M_2	M_3	M_4	M_5	M_6
$\mathcal{P}_1^{\mathcal{DM}}$	56	5	1	0	0.13		29	0	12	0	0.58		51	0	13	0	0.27		24	0	12	0	0.66		0	0	0	0	1	10
$\mathcal{P}_1^{\mathcal{DH}_3}$	42	5	1	0	0.33		4	0	0	0	0.94		38	2	7	2	0.43		0	2	0	2	0.97	24	–	–	–	–	–	–
$\mathcal{P}_1^{\mathcal{DH}_2}$	12	5	0	0	0.76		6	0	0	0	0.91		0	2	0	2	0.97	64	–	–	–	–	–	–	–	–	–	–	–	–
$\mathcal{P}_1^{\mathcal{DH}_1}$	39	5	1	0	0.37		0	4	0	4	0.94	27	38	2	7	2	0.43		–	–	–	–	–	–	–	–	–	–	–	–
$\mathcal{P}_1^{\mathcal{NR}}$	0	0	0	0	1	70	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
$\mathcal{P}_2^{\mathcal{DM}}$	64	14	2	0	0.11		27	0	24	0	0.69		60	0	26	0	0.32		23	0	22	0	0.74		0	0	0	0	1	12
$\mathcal{P}_2^{\mathcal{DH}_3}$	57	14	1	0	0.19		2	0	0	0	0.97		54	3	21	3	0.35		0	3	0	3	0.97	19	–	–	–	–	–	–
$\mathcal{P}_2^{\mathcal{DH}_2}$	16	14	2	0	0.66		6	0	4	0	0.93		0	3	0	3	0.97	62	–	–	–	–	–	–	–	–	–	–	–	–
$\mathcal{P}_2^{\mathcal{DH}_1}$	55	14	1	0	0.21		0	11	0	11	0.88	21	54	3	21	3	0.35		–	–	–	–	–	–	–	–	–	–	–	–
$\mathcal{P}_2^{\mathcal{NR}}$	0	0	0	0	1	88	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
$\mathcal{P}_3^{\mathcal{DM}}$	75	13	6	0	0.08		47	0	31	0	0.51		69	0	35	0	0.28		41	0	27	0	0.57		0	0	0	0	1	14
$\mathcal{P}_3^{\mathcal{DH}_3}$	58	13	4	0	0.26		6	0	4	0	0.94		46	2	15	2	0.50		0	2	0	2	0.98	31	–	–	–	–	–	–
$\mathcal{P}_3^{\mathcal{DH}_2}$	22	13	4	0	0.64		6	0	4	0	0.94		0	2	0	2	0.98	70	–	–	–	–	–	–	–	–	–	–	–	–
$\mathcal{P}_3^{\mathcal{DH}_1}$	55	13	3	0	0.29		0	7	0	7	0.93	34	46	2	15	2	0.50		–	–	–	–	–	–	–	–	–	–	–	–
$\mathcal{P}_3^{\mathcal{NR}}$	0	0	0	0	1	96	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 2: Comparison of Access Control Models

Results.

Table 2 provides an overview of the results. For each policy set $\mathcal{P}_i^j$ (with $i \in \{1, \dots, 3\}$ and $j \in \{NR, \mathcal{DH}_1, \mathcal{DH}_2, \mathcal{DH}_3, \mathcal{DM}\}$), the table reports the metrics M_1 to M_6 computed when the policy set is evaluated with respect to $\mathcal{AC}_{\mathcal{NR}}$, $\mathcal{AC}_{\mathcal{DH}_1}$, $\mathcal{AC}_{\mathcal{DH}_2}$, $\mathcal{AC}_{\mathcal{DH}_3}$ and $\mathcal{AC}_{\mathcal{DM}}$. Note that we did not evaluate $\mathcal{P}_i^{\mathcal{DH}_3}$ within $\mathcal{AC}_{\mathcal{DM}}$; $\mathcal{P}_i^{\mathcal{DH}_1}$ and $\mathcal{P}_i^{\mathcal{DH}_2}$ within $\mathcal{AC}_{\mathcal{DH}_3}$ and $\mathcal{AC}_{\mathcal{DM}}$; and $\mathcal{P}_i^{\mathcal{NR}}$ within $\mathcal{AC}_{\mathcal{DH}_1}$, $\mathcal{AC}_{\mathcal{DH}_2}$, $\mathcal{AC}_{\mathcal{DH}_3}$ and $\mathcal{AC}_{\mathcal{DM}}$. This is because $\mathcal{AC}_{\mathcal{DM}}$ has full coverage (i.e., $M_5 = 1$) for $\mathcal{P}_i^{\mathcal{DM}}$, $\mathcal{AC}_{\mathcal{DH}_3}$ has full coverage of $D_{Int}^{\mathcal{P}_i^-}$ for $\mathcal{P}_i^{\mathcal{DH}_3}$, $\mathcal{AC}_{\mathcal{DH}_2}$ has full coverage of $D_{Int}^{\mathcal{P}_i^-}$ for $\mathcal{P}_i^{\mathcal{DH}_2}$, and $\mathcal{AC}_{\mathcal{DH}_1}$ has full coverage of $D_{Int}^{\mathcal{P}_i^-}$ for $\mathcal{P}_i^{\mathcal{DH}_1}$.

We can observe in the table that $\mathcal{AC}_{\mathcal{DM}}$ achieves the best trade-off between *coverage* (M_5) and *efficiency* (M_6). In fact, $\mathcal{AC}_{\mathcal{DM}}$ covers the set of intended protected data with a significantly less number of policies than $\mathcal{AC}_{\mathcal{NR}}$, $\mathcal{AC}_{\mathcal{DH}_1}$, $\mathcal{AC}_{\mathcal{DH}_2}$ and $\mathcal{AC}_{\mathcal{DH}_3}$. For instance, $\mathcal{P}_1^{\mathcal{NR}}$ consists of 70 policy statements, $\mathcal{P}_1^{\mathcal{DH}_1}$ of 27 statements, $\mathcal{P}_1^{\mathcal{DH}_2}$ of 64 statements and $\mathcal{P}_1^{\mathcal{DH}_3}$ of 24 statements, while $\mathcal{P}_1^{\mathcal{DM}}$ is formed by only 10 policy statements. In particular, to define a policy that covers the full set of intended protected data in $\mathcal{AC}_{\mathcal{NR}}$, a user has to define a policy for each data element in the set. However, the user can only do that if he has a deep knowledge of the application domain, which is rarely the case for an average user. In contrast, $\mathcal{AC}_{\mathcal{DM}}$ relieves users of this burden by relying on a data model encoding the domain knowledge for decision making. Moreover, the results show that $\mathcal{AC}_{\mathcal{DH}_2}$ requires the specification of a policy for a larger number of data elements compared to $\mathcal{AC}_{\mathcal{DH}_1}$ and $\mathcal{AC}_{\mathcal{DH}_3}$ in order to prevent the leakage of sensitive information. This is due to the fact that, when using $\mathcal{AC}_{\mathcal{DH}_2}$, policies have to be defined for leaf nodes, and in a data hierarchy the number of nodes in a stratum is usually larger than the number of nodes in the stratum immediately above it.

It is worth noting that it may not be possible to obtain full coverage of the set of intended protected data within $\mathcal{AC}_{\mathcal{DH}_1}$, $\mathcal{AC}_{\mathcal{DH}_2}$ and $\mathcal{AC}_{\mathcal{DH}_3}$, i.e. there may not exist policy sets $\mathcal{P}_i^{\mathcal{DH}_1}$, $\mathcal{P}_i^{\mathcal{DH}_2}$ and $\mathcal{P}_i^{\mathcal{DH}_3}$ respectively, such that $M_5 = 1$. In order to define a policy capturing their privacy constraints, users should know the data elements that allow the inference of the data elements to which they wants to restrict the access and define negative authorizations for those data elements explicitly. However, these negative authorizations are not correctly interpreted by $\mathcal{AC}_{\mathcal{DH}_2}$ and $\mathcal{AC}_{\mathcal{DH}_3}$. In particular, they are propagated both up and down the hierarchy in $\mathcal{AC}_{\mathcal{DH}_3}$ and up the hierarchy in $\mathcal{AC}_{\mathcal{DH}_2}$, thus restricting the access to data elements which the user wants to disclose. On the other hand, when a negative authorization is defined for a data element, $\mathcal{AC}_{\mathcal{DH}_1}$ requires the specification of a negative authorization for the root elements of the data hierarchy. This, by propagation, might prevent access to data elements for which access should be allowed. Consequently, policies expressed in access control models only relying on a data hierarchy, result to be more restrictive than what the user aims at as indicated by M_4.

The experiments also show that $\mathcal{AC}_{\mathcal{NR}}$, $\mathcal{AC}_{\mathcal{DH}_1}$, $\mathcal{AC}_{\mathcal{DH}_2}$ and $\mathcal{AC}_{\mathcal{DH}_3}$ do not fully protect a user from data leakages without a comprehensive knowledge of the application domain. This becomes evident when we evaluate $\mathcal{P}_i^{\mathcal{DM}}$ with respect to these models. When $\mathcal{P}_i^{\mathcal{DM}}$ is evaluated within $\mathcal{AC}_{\mathcal{DH}_1}$, $\mathcal{AC}_{\mathcal{DH}_2}$ and $\mathcal{AC}_{\mathcal{DH}_3}$, there may be situations in which the user would like to deny access but these models do not (M_1), or even worse, situations in which the access should be denied while $\mathcal{AC}_{\mathcal{DH}_1}$, $\mathcal{AC}_{\mathcal{DH}_2}$ and $\mathcal{AC}_{\mathcal{DH}_3}$ grant access (M_3), thus allowing inference of user sensitive data. Indeed, the propagation of positive authorizations through hierarchies increases the number of data elements for which permission is allowed: those permissions grant access to user's information that should be protected, leading to leakages of sensitive information. Similar situations happen when $\mathcal{P}_i^{\mathcal{DM}}$ is evaluated with respect to $\mathcal{AC}_{\mathcal{NR}}$. Since the user relies on her (limited) knowledge of the application domain to define a policy set that covers the whole set of intended protected data, the defined policies might not reflect her constraints on the disclosure of sensitive information. Indeed, there are several cases where the user would like to deny access but $\mathcal{AC}_{\mathcal{NR}}$ does not (M_1) or vice versa (M_2). Moreover, a user may explicitly grant access to a data element that allows the inference of a data element to which the user has explicitly denied the access (M_3). Comparing $\mathcal{AC}_{\mathcal{NR}}$, $\mathcal{AC}_{\mathcal{DH}_1}$, $\mathcal{AC}_{\mathcal{DH}_2}$ and $\mathcal{AC}_{\mathcal{DH}_3}$ with respect to the evaluation of $\mathcal{P}_i^{\mathcal{DM}}$, we can observe that, in general, $\mathcal{AC}_{\mathcal{DH}_3}$ better captures a user's intention than $\mathcal{AC}_{\mathcal{DH}_1}$ and $\mathcal{AC}_{\mathcal{DH}_2}$; in turn, $\mathcal{AC}_{\mathcal{DH}_1}$ and $\mathcal{AC}_{\mathcal{DH}_2}$ better capture a user's intention than $\mathcal{AC}_{\mathcal{NR}}$ (M_1 and M_2). However, $\mathcal{AC}_{\mathcal{DH}_1}$, $\mathcal{AC}_{\mathcal{DH}_2}$ and $\mathcal{AC}_{\mathcal{DH}_3}$ are more prone to data leakages when a user does not have a sufficient knowledge of the application domain and, in particular, of inference relations (M_3).

Note that $\mathcal{AC}_{\mathcal{DH}_2}$ can achieve the same level of accuracy offered by $\mathcal{AC}_{\mathcal{DH}_3}$. This can be explained by the fact that the it is always possible to define a policy $\mathcal{P}_i^{\mathcal{DH}_2}$ which evaluated using $\mathcal{AC}_{\mathcal{DH}_2}$

behaves as a policy $\mathcal{P}_i^{\mathcal{DH}_3}$ evaluated using $\mathcal{AC}_{\mathcal{DH}_3}$. However, such a policy requires a larger amount of policy statements compared to the corresponding policy for $\mathcal{AC}_{\mathcal{DH}_3}$. On the other hand, the maximum level of accuracy that can be reached by $\mathcal{AC}_{\mathcal{DH}_1}$ (i.e., for $\mathcal{P}_i^{\mathcal{DH}_1}$) is lower than the maximal accuracy that can be reached by $\mathcal{AC}_{\mathcal{DH}_2}$ and $\mathcal{AC}_{\mathcal{DH}_3}$. Moreover, one can observe that $\mathcal{P}_i^{\mathcal{DH}_1}$ and $\mathcal{P}_i^{\mathcal{DH}_3}$ behave in a similar way with respect to $\mathcal{AC}_{\mathcal{DH}_2}$. This can be explained by the fact that policy $\mathcal{P}_i^{\mathcal{DH}_1}$ can be seen as an extension of $\mathcal{P}_i^{\mathcal{DH}_3}$. In other words, it can be obtained from $\mathcal{P}_i^{\mathcal{DH}_3}$ by adding the statements needed to reduce the number of cases in access is not properly denied (i.e., M_1).

Based on these observations, we can conclude that $\mathcal{AC}_{\mathcal{DM}}$ performs better than existing access control models in that it provides full protection from data leakages with lower efforts on the user side. In particular, it minimizes the number of policies a user has to write while allowing full coverage of the intended protected set.

7. RELATED WORK

In this paper we have investigated the problem of preventing inference of sensitive information in access control. Inference control aims to prevent indirect access to sensitive information where a user learns sensitive information from non-sensitive one. Access control, instead, prevents direct access to sensitive information.

In the remainder of this section, we discuss approaches that focus on inference control and approaches that focus on the combination of inference control and access control.

Inference Control.

Several approaches to inference control have been proposed in the literature [1, 15, 18, 37], especially for database systems [3, 7]. The inference problem in databases occurs when sensitive information is disclosed indirectly by combining the answers to a sequence of non-sensitive queries. For this reason, most of the proposed approaches focus on controlling query execution at runtime. For instance, Biskup and Bonatti [7] propose a technique for controlled query evaluation, which modifies the ordinary query evaluation by distorting answers if necessary to preserve confidentiality with respect to a given confidentiality policy.

However, these techniques are computationally inefficient. Thus, Biskup et al. [8, 9, 10] propose to reduce the problem of inference control to access control. These approaches are based on introducing constraints which represent a combination of attribute values in a tuple to be kept secret. If the constants in a query match the constraints, the query asks for a secret value and therefore is not allowed. These approaches are complementary to our approach because they consider inference deriving from the combination of different attribute values. In our work instead we tackle the inference problem by adopting a semantic approach in which inference is prevented on the basis of hierarchical and inference relations between data elements.

Inference and Access Control.

Only few works have considered the issue of information inference in access control [25, 34]. For instance, Katos et al. [25] present an approach based on the concept of inference control by design. They model inference channels between attributes in a data schema as a disclosure matrix that, for each attribute, represents the probability of an attribute revealing other attributes. This matrix is used to compute an access leakage matrix capturing the effective access control. The access leakage matrix is used to define an access control policy that avoids inference channels using separation of duty constraints. However, inference is only analyzed at the attribute level, thus limiting the granularity of the analysis. In contrast, our approach enables to reason on information inference at any level of the data hierarchy. In particular, we have tackled the problem of inference control by proposing an access control model that adopts semantic inference relationship among data and defines on top of these relationship authorization propagation rules that prevent inference of sensitive information.

The key role played by semantic relationships among data in making access control decisions has been recognized in earlier work [12, 34, 41]. Crampton and Sellwood [12] propose a generic access control model using relationships among entities as the basis to specifying authorization rules. An access request is evaluated with respect to two types of rules: principal matching rules and authorization rules. The former express conditions on the relationships that form a path between a subject s and an object o in an access request (s, o, a); the latter specify the actions that the subject s can/cannot perform on the object o. Similarly, in our approach we consider (semantic) relations between data as the basis of the evaluation of authorization policies. These relations can be considered as a special case of the relations among entities used in [12] to define principal matching rules.

Vavilis et al. [41] use inference relations to reason on data sensitivity and quantify the severity of data leakages. Similarly to [41], our approach is based on a data model which considers both hierarchical and inference relations between data elements. Leveraging these two types of relations makes it possible to detect situations in which access control policies permit access to data from which it is possible to infer sensitive information the user would not like to disclose.

The closest work to ours is the one of Qin et al. [34]. This work presents an access control model for Semantic Web, which allows the specification of authorizations over concepts defined in ontologies and their enforcement upon data instances annotated by concepts. The model makes use of semantic relationships among concepts to define authorization propagation policies that prevent inference of sensitive information. The authors also show how policies can be represented in an OWL-based access control language.

As in [34], we use domain knowledge (possibly represented using an ontology) and, in particular, the semantic relationship between data elements to define propagation rules that prevent leakage of sensitive information. However, there are significant differences between our work and the one in [34]. First, we discriminate between hierarchical relations and inference relations in the definition of authorization propagation rules while Qin et al. do not make this distinction. In particular, Qin et al. classify the relations that can occur in an ontology (including hierarchical relations) in three categories, namely inferable, partially inferable and non-inferable. Based on this classification they propose four propagation rules. These rules are used to propagate both positive and negative authorizations regardless of the type of relations (except non-inferable relations that do not propagate authorizations). In contrast, the definition of our propagation rules is driven by the observation that granting access to a data element does not imply granting access also to the data elements from which it can be inferred. As a consequence, the propagation rules proposed in [34] allow the access to information to a wider audience than intended. Moreover, Qin et al. propose their own language based on OWL to express access control policies, which requires the development of an ad-hoc engine for policy evaluation. In contrast, we encode policies and data model in XACML, thus making our model directly implementable using any existing XACML-compliant engine. More importantly, OWL-based policy languages have inherent limitations in the expressiveness of the policies that can be enforced due to the fact that

OWL with rules is undecidable if unrestricted [26]. On the other hand, our approach can leverage the expressivity and extensibility of XACML for the specification of access control policies, resulting in a larger set of policies that can be supported by our access control model.

8. CONCLUSION

In this paper we have presented an access control model that uses a semantic approach to prevent inference of sensitive information. The model is based on a data model that encodes domain knowledge. In particular the data model organizes domain knowledge in a hierarchical structure and makes inference relations between data explicit. These relations are used to define authorization propagation rules that prevent inference of sensitive information.

We acknowledge that the proposed access control model can only prevent inference of sensitive information based on the inference relations that are explicit in the data model. Thus, our approach is complementary to non-semantic approaches to inference control that consider inference of sensitive information as combination of non-sensitive one.

We have evaluated and compared the proposed access control model with existing access control models which either do not consider relations among data or rely only on data hierarchies to determine applicable policies. The experiments show that our model overcomes the limitations of existing access control models in that it provides protection against explicit secondary data leakages and reduces the effort required by a user to specify a policy expressed in term of number of statements that the user has to specify.

The work presented in this paper poses the basis for several directions for future work. The aim of the proposed access control model is to prevent inference of sensitive information. This, however, may result in users not being fully conscious about the side effects that defining a certain policy may lead to. To this end, the access control model can be complemented with transparency tools that help users in the understanding of the consequences of the defined policies. Transparency tools can be used either to analyze users' policies at design time, for instance based on policy analysis approaches [20, 40], or at run time to provide feedback to users when the access decision differs from the authorizations they have explicitly specified in their policies [14].

Moreover, we are planning to conduct an extensive evaluation of our approach with respect to different aspects. First, we want to evaluate the effectiveness and efficiency of our approach in a controlled experiment in which users specify an access control policy using the proposed access control model, an access control model that do not consider relations among data and one that makes use of data hierarchies. Second, since the proposed access control model relies on the representation of the application domain for the decisional process, we want to study the impact of the representation of the application domain has on the effectiveness of the access control model. To this end, we plan to perform experiments in which different domain specific ontologies like FOAF for social networks, GoodRelations [19] and CContology [24] for e-commerce are used to instantiate the data model.

Acknowledgments

This work has been partially funded by the ITEA2 projects FedSS and M2MGrid, the EDA project IN4STARS2.0, the Dutch national program COMMIT under the THeCS project, the EU under grant agreement n.285223 (SECONOMICS) and the SESAR JU WPE under contract 12-120610-C12 (EMFASE).

9. REFERENCES

[1] R. Accorsi and G. Muller. Preventive inference control in data-centric business models. In *Proceedings of Security and Privacy Workshops*, pages 28–33. IEEE, 2013.

[2] A. Acquisti and R. Gross. Imagined communities: Awareness, information sharing, and privacy on the facebook. In *Privacy Enhancing Technologies*, LNCS 4258, pages 36–58. Springer, 2006.

[3] K. Arkoudas and A. Vashist. A model-theoretic approach to data anonymity and inference control. In *Proceedings of Conference on Data and Application Security and Privacy*, pages 249–256. ACM, 2012.

[4] M. Backes, G. Karjoth, W. Bagga, and M. Schunter. Efficient comparison of enterprise privacy policies. In *Proceedings of Symposium on Applied Computing*, pages 375–382. ACM, 2004.

[5] A. Barth and J. C. Mitchell. Enterprise privacy promises and enforcement. In *Proceedings of Workshop on Issues in the Theory of Security*, pages 58–66. ACM, 2005.

[6] E. Bertino, P. Samarati, and S. Jajodia. An extended authorization model for relational databases. *IEEE Trans. on Knowl. and Data Eng.*, 9(1):85–101, 1997.

[7] J. Biskup and P. Bonatti. Controlled query evaluation for enforcing confidentiality in complete information systems. *International Journal of Information Security*, 3(1):14–27, 2004.

[8] J. Biskup, D. W. Embley, and J. Lochner. Reducing inference control to access control for normalized database schemas. *Inf. Process. Lett.*, 106(1):8–12, 2008.

[9] J. Biskup, S. Hartmann, S. Link, and J.-H. Lochner. Efficient inference control for open relational queries. In *Data and Applications Security and Privacy XXIV*, LNCS 6166, pages 162–176. Springer, 2010.

[10] J. Biskup and J.-H. Lochner. Enforcing confidentiality in relational databases by reducing inference control to access control. In *Information Security*, LNCS 4779, pages 407–422. Springer, 2007.

[11] J.-W. Byun and N. Li. Purpose based access control for privacy protection in relational database systems. *The VLDB Journal*, 17(4):603–619, 2008.

[12] J. Crampton and J. Sellwood. Path conditions and principal matching: A new approach to access control. In *Proceedings of Symposium on Access Control Models and Technologies*, pages 187–198. ACM, 2014.

[13] M. J. Culnan and P. K. Armstrong. Information privacy concerns, procedural fairness, and impersonal trust: An empirical investigation. *Organization Science*, 10(1):04–115, 1999.

[14] S. Damen, J. den Hartog, and N. Zannone. CollAC: Collaborative access control. In *Proceedings of International Conference on Collaboration Technologies and Systems*, pages 142–149. IEEE, 2014.

[15] J. Domingo-Ferrer. A Survey of Inference Control Methods for Privacy-Preserving Data Mining. In *Privacy-Preserving Data Mining*, Advances in Database Systems 34, pages 53–80. Springer, 2008.

[16] M. Egea, F. Paci, M. Petrocchi, and N. Zannone. PERSONA - A Personalized Data Protection Framework. In *Proceedings of IFIP WG 11.11 International Conference on Trust Management*, IFIP Advances in Information and Communication Technology 401, pages 272–280. Springer, 2013.

[17] P. Guarda and N. Zannone. Towards the development of privacy-aware systems. *Information & Software Technology*, 51(2):337–350, 2009.

[18] R. Heatherly, M. Kantarcioglu, and B. M. Thuraisingham. Preventing private information inference attacks on social networks. *IEEE Trans. Knowl. Data Eng.*, 25(8):1849–1862, 2013.

[19] M. Hepp. GoodRelations: An Ontology for Describing Products and Services Offers on the Web. In *Proceedings of International Conference on Knowledge Engineering: Practice and Patterns*, LNCS 5268, pages 329–346. Springer, 2008.

[20] G. Hughes and T. Bultan. Automated Verification of Access Control Policies Using a SAT Solver. *Int. J. Softw. Tools Technol. Transf.*, 10(6):503–520, 2008.

[21] IHTSDO. SNOMED CT – The Global Language of Healthcare. http://www.ihtsdo.org/snomed-ct.

[22] ISO/IEC 25010:2011. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2011.

[23] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001.

[24] M. Jarrar. *Towards Effectiveness and Transparency in e-Business Transactions, An Ontology for Customer Complaint Management*, chapter 7. Idea Group Inc., 2009.

[25] V. Katos, D. Vrakas, and P. Katsaros. A framework for access control with inference constraints. In *Proceedings of Computer Software and Applications Conference*, pages 289–297. IEEE, 2011.

[26] M. Krötzsch, S. Rudolph, and P. Hitzler. Description logic rules. In *Proceedings of European Conference on Artificial Intelligence*, Frontiers in Artificial Intelligence and Applications 178, pages 80–84. IOS Press, 2008.

[27] J.-G. Lee, K.-Y. Whang, W.-S. Han, and I.-Y. Song. The dynamic predicate: integrating access control with query processing in XML databases. *The VLDB Journal*, 16(3):371–387, 2007.

[28] A. Masoumzadeh and J. Joshi. PuRBAC: Purpose-Aware Role-Based Access Control. In *On the Move to Meaningful Internet Systems*, LNCS 5332, pages 1104–1121. Springer, 2008.

[29] I. Matteucci, P. Mori, and M. Petrocchi. Prioritized Execution of Privacy Policies. In *Data Privacy Management and Autonomous Spontaneous Security*, LNCS 7731, pages 133–145. Springer, 2012.

[30] G. R. Milne and M.-E. Boza. Trust and concern in consumers' perceptions of marketing information management practices. *Journal of Interactive Marketing*, 13(1):5 – 24, 1999.

[31] OASIS XACML Technical Committee. XACML v3.0 Core and Hierarchical Role Based Access Control (RBAC) Profile Version 1.0. Committee specification, OASIS, 2010.

[32] OASIS XACML Technical Committee. XACML v3.0 Hierarchical Resource Profile Version 1.0. Committee specification, OASIS, 2010.

[33] OASIS XACML Technical Committee. eXtensible Access Control Markup Language (XACML) Version 3.0. Oasis standard, OASIS, 2013.

[34] L. Qin and V. Atluri. Concept-level access control for the semantic web. In *Proceedings of ACM Workshop on XML Security*, pages 94–103. ACM, 2003.

[35] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. A model of authorization for next-generation database systems. *ACM Trans. Database Syst.*, 16(1):88–131, 1991.

[36] C. Ruan and S. Shahrestani. Logic based authorization program and its implementation. In *Proceedings of International Conference on Security of Information and Networks*, pages 87–94. ACM, 2011.

[37] A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2006.

[38] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[39] A. Squicciarini, F. Paci, and S. Sundareswaran. PriMa: a comprehensive approach to privacy protection in social network sites. *Annales des Télécommunications*, 69(1-2):21–36, 2014.

[40] F. Turkmen, J. den Hartog, S. Ranise, and N. Zannone. Analysis of XACML Policies with SMT. In *Principles of Security and Trust*, LNCS 9036, pages 115–134. Springer, 2015.

[41] S. Vavilis, M. Petkovic, and N. Zannone. Data leakage quantification. In *Data and Applications Security and Privacy*, LNCS 8566, pages 98–113. Springer, 2014.