Tuple-Based Access Control: a Provenance-Based Information Flow Control for Relational Data

Romuald Thion, François Lesueur, Meriam Talbi

Tuple-Based Access Control: a Provenance-Based Information Flow Control for Relational Data

Romuald Thion
Université Lyon 1, CNRS
LIRIS, UMR5205
romuald.thion@univ-lyon1.fr

François Lesueur
INSA-Lyon, CNRS
LIRIS, UMR5205
francois.lesueur@insa-lyon.fr

Meriam Talbi
INSA-Lyon, CNRS
LIRIS, UMR5205
meriam.talbi@insa-lyon.fr

September 3, 2015

Abstract

This paper proposes a flexible control framework for relational personal data that enforces data originators' dissemination policies. Inspired by the sticky policy paradigm and mandatory access control, dissemination policies are linked with atomic data and are combined when different pieces of data are merged. The background setting of relational provenance guarantees that the policy combining operations behave accordingly to the operations carried out on the data. We show that the framework can capture a large class of policies similar to those of lattice-based access control models and that it can be integrated seamlessly into relational database management systems. In particular, we define a path oriented dissemination control model where policies define authorized chains of transfers between databases.

Promising ongoing research work include the generalization of the theoretical framework to more expressive query languages including aggregation and difference operators as well as experiments on secure tokens.

Keywords: access control, relational databases, provenance, information flow, personal data server.

1 Introduction

The digitization of personal files has proved to be a convenient procedure to store, find, query and transfer these files. Its advantages have led to the large amount of digitized personal data we own today. Since such data contain private information by nature, it is mandatory to offer document and file owners some control over the diffusion of their personal data. Secure Personal Data Servers (PDSs) built upon portable and secure devices have emerged as a possible technical solution to deal with the issues raised by the proliferation of personal data by providing secure storage. A PDS is a USB-sized token equipped with a tiny processor, cryptographic capabilities and protection against physical tampering. It has been shown that a Relational DataBase Management System (R-DBMS) can be embedded into a secure smart card, to provide a real breakthrough in the management of sensitive data [3]. However access control and information flow control solutions need to be developed for such environments and for personal relational databases as well.

This paper defines an information flow control model, namely *Tuple-Based Access Control* (TBAC), inspired by lattice-based and Mandatory Access Control (MAC) models [23] as well as by the sticky policy paradigm [12] and Hippocratic DBMS [1]. The objective of TBAC is to provide a mechanism that controls

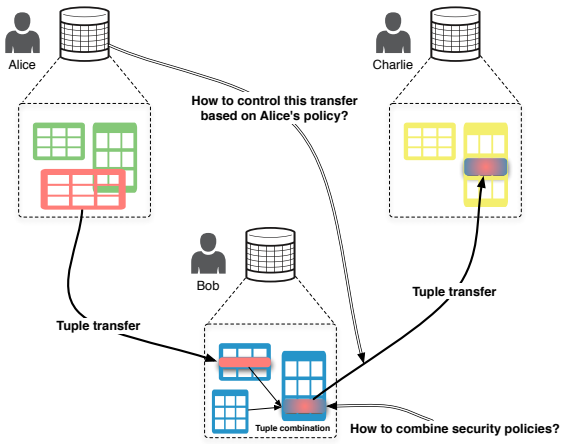

Figure 1: Data transfer between PDS

the dissemination of tuples between R-DBMS according to the authorizations defined by the producers of the initial tuples. TBAC authorization policies are attached to tuples as a specific form of provenance information, then, when tuples are accessed by means of relational queries, the related policies are combined accordingly to the queries' structures. In this paper we emphasize PDS-based applications, but the TBAC approach is more generally applicable to relational databases.

Figure 1 illustrates a typical data transfer scenario between three participants that own personal relational databases powered by PDSs: the PDSs both act as clients that send queries to other participants' PDSs as well as servers that compute queries and send results to requesters. The point of TBAC is to filter results in such a way that tuples owners' policies are enforced along chains of transfers. In the scenario of Figure 1, Alice is the initial producer who tags each of her tuples at insertion time with an authorization policy that stipulates who has access to her personal information. Bob, one of Alice's colleagues, executes a query on Alice's PDS. Bob does not receive the complete answer to his query but only a subset of it, filtered according to Bob's credentials. Then Charlie asks Bob for Alice's data combined with others' tuples under Bob's sovereignty: at this point, we aim at enforcing an access control complying with both the access rights Alice initially put on her tuples as well as the access rights stipulated by Bob on his ones. Doing so, we can control the transfer of information between the participants of the system made of all PDSs with the guarantee that the initial will of tuple producers are satisfied.

In Section 2, we provide background information on the provenance model on which TBAC relies. In Section 3 we define the TBAC access-control models family: a flexible tuple-grained model that integrates with the core engine of a R-DBMS. More precisely, we define an abstract settings, give its semantics and exemplify it with several kinds of authorization policies. The two key differences between TBAC and traditional MAC in R-DBMS (e.g., Oracle Label Security) are the treatment of alternation and the seamless integration of control into query evaluation. In Section 4, we focus on dissemination control to show how to express authorized path policies that control where data can flow from one PDS to another. In Section 5, we compare TBAC against standard access control models used in R-DBMS and we survey related work on dissemination control and provenance-based access control models. Finally, in Section 6, we discuss the design of TBAC, its usability and we show how to use Ciphertext-Policy Attribute Based Encryption to enforce TBAC, before concluding this paper and suggesting further research work in Section 7.

2 Relational provenance

In TBAC, security policies are labels called *s-tags*, for *security tags*, associated to tuples, as are labels in MAC models. However, we do not assume that labels are automatically deduced from the users' credentials

r	A	B	C	*s-tag*
t_0	a	b	c	k_0
t_1	d	b	e	k_1
t_2	f	g	e	k_2

$q(r)$	A	C	*s-tag*
s_0	a	c	$(k_0 \otimes k_0) \oplus (k_0 \otimes k_0)$
s_1	a	e	$(k_0 \otimes k_1)$
s_2	d	c	$(k_0 \otimes k_1)$
s_3	d	e	$(k_1 \otimes k_1) \oplus (k_1 \otimes k_1) \oplus (k_1 \otimes k_2)$
s_4	f	e	$(k_2 \otimes k_2) \oplus (k_2 \otimes k_2) \oplus (k_1 \otimes k_2)$

Table 1: Instance r and result $q(r)$

but are rather user-defined statements. The main building block to implement a sticky policy approach is a mechanism that deals with *s-tags* consistently with the operations carried out on the tuples. Green, Karvounarakis and Tannen have introduced a formal framework – the *extended Relational Algebra* ($\mathcal{RA}^+$) – that provides a positive and elegant answer to this problem [13] in the setting of the *positive relational algebra* (Selection (σ), Projection (π), Join ($\bowtie$), Renaming (ρ) and Union ($\cup$)). In the $\mathcal{RA}^+$ model each tuple is annotated with a value from a set $\mathbb{K}$ of *s-tags*. When tuples are combined by SPJRU expressions their related metadata are combined accordingly using two operations: $\oplus : \mathbb{K} \times \mathbb{K} \to \mathbb{K}$ and $\otimes : \mathbb{K} \times \mathbb{K} \to \mathbb{K}$ that act as generalized versions of the boolean operators $\vee$ and $\wedge$ used in the classical relational algebra with set semantics. Informally, $\oplus$ combines *s-tags* of tuples involved in unions and projections while $\otimes$ is used by joins and cartesian product operations.

Relational algebra obeys some identities (e.g., $R \bowtie (S \cup T) = R \bowtie S \cup R \bowtie T$), a key result of $\mathcal{RA}^+$ that we will use in Section 3 is to show that the algebraic structures that reflect and preserve fundamental identities of the positive relational algebra are precisely *commutative semirings*. A commutative semiring on an underlying set $\mathbb{K}$ is an algebraic variety $\langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle$ made of two different binary operations $\oplus$ and $\otimes$ with two distinguished elements 0 and 1. The substructures $\langle \mathbb{K}, \oplus, 0 \rangle$ and $\langle \mathbb{K}, \otimes, 1 \rangle$ are both *commutative monoids*, with the additional properties that $\otimes$ *distributes* over $\oplus$ and that 0 is *absorbing* for $\otimes$. Thus, it is basically a distributive lattice without the idempotency requirement. A *homomorphism* (of semirings) between $\langle \mathbb{K}_1, \oplus_1, \otimes_1, 0_1, 1_1 \rangle$ and $\langle \mathbb{K}_2, \oplus_2, \otimes_2, 0_2, 1_2 \rangle$ is a function $f : \mathbb{K}_1 \to \mathbb{K}_2$ that satisfies the usual requirements of a structure-preserving map, $f(x \oplus_1 y) = f(x) \oplus_2 f(y)$, $f(x \otimes_1 y) = f(x) \otimes_2 f(y)$, $f(0_1) = 0_2$, $f(1_1) = 1_2$. Details are to be found in [13].

As an example, consider the instance r given by Table 1 which contains three tuples annotated respectively with k_0, k_1 and k_2. We assume that the instance r is held by Alice in her PDS. Bob runs the following SPJRU query q on Alice's secure PDS: $q(r) = \pi_{AC}(\pi_{AB}(r) \bowtie \pi_{BC}(r) \cup \pi_{AC}(r) \bowtie \pi_{BC}(r))$ Table 1 illustrates the $\mathcal{RA}^+$ semantics, for example, s_3 's annotation $(k_1 \otimes k_1) \oplus (k_1 \otimes k_1) \oplus (k_1 \otimes k_2)$ indicates that there are three different ways to obtain (d, e), one for each monomial: by joining t_1 with itself on B (first monomial $k_1 \otimes k_1$); by joining t_1 with itself on C (second monomial $k_1 \otimes k_1$); by combining t_1 with t_2 (monomial $k_1 \otimes k_2$).

3 The TBAC models family

The key idea behind the design of the TBAC model is to think about a semiring on $\mathbb{K}$ as the algebraic structure of security policies, similar to the lattices used in lattice-based access control and MAC. The first ingredient is the semiring $\langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle$ of *access rights* that are attached to tuples. The two distinguished elements $0 \in \mathbb{K}$ and $1 \in \mathbb{K}$ represent the *deny-all* and the *allow-all s-tag* respectively. Operators $\oplus$ and $\otimes$ are policy combinators that captures summation (a.k.a., disjunction) and product (a.k.a., conjunction) of policies. The second ingredient is the set $\mathbb{C}$ of *credentials* that are associated with users. The last ingredient is the decision function $f : \mathbb{C} \times \mathbb{K} \to \mathbb{B}$ (with $\mathbb{B} = \{\top, \bot\}$) that decides whether a tuple with *s-tag* $k \in \mathbb{K}$ should be authorized or not by comparing it against the requester's credentials $c \in \mathbb{C}$. The decision function captures the access control semantics of TBAC: a user with credentials c has access to a tuple t, tagged with k, in a query result $q(r)$ only if the decision function $f(c)(k)$ is $\top$. In the rest of this section we introduce

motivating instances of the framework, entitled *UserSet*, *AttributeSet* and *Deadline* TBAC, we show the core semantic property of the framework and prove that different TBAC policies can be integrated. From now on, we will write $(t : k)$ for a tuple t and its *s-tag* k.

3.1 Example instances

3.1.1 UserSet TBAC

In *UserSet* TBAC a user is allowed to access a tuple based on his/her identity and the set of authorized users. We note by U the set of all the users in the system. Each tuple t is annotated by an element $k \subseteq U$, which means that every user in k is allowed to read t. Formally, we have $\mathbb{K} = \mathcal{P}(U)$ with $\mathcal{P}$ the powerset operator and $\mathbb{C} = U$. UserSet TBAC amounts to instantiating the $\mathcal{RA}^+$ framework by choosing the structure $\langle \mathcal{P}(U), \cup, \cap, \emptyset, U \rangle$ as the domain of *s-tags*. For a user u requesting a tuple $(t : k)$, access is granted if $u \in k$. The decision function is thus defined by $f(c)(k) = c \in k$.

3.1.2 AttributeSet TBAC

We can go a bit further to capture attribute-based access control. In *AttributeSet* TBAC an *s-tag* is given as *a set of sets of authorized attributes*, that is an *s-tag* is an element of $\mathcal{P}(\mathcal{P}(G))$ where G denotes the set of all attributes. The access control semantics is the following: the outer set level captures alternation, that is, an access to t is authorized if it is granted by $\{g_0^0, \ldots, g_{l(0)}^0\}$ *or* $\ldots$ *or* by $\{g_0^n, \ldots, g_{l(n)}^n\}$; the inner set level captures conjunction, that is, an access is granted if *all* the g_j^i of one group are satisfied. The operation that captures the addition of policies is naturally $\oplus = \cup$. Multiplication of policies is formally defined by the operation $X \Cup Y = \{x \cup y \mid x \in X \wedge y \in Y\}$. The resulting semiring is $\langle \mathcal{P}(\mathcal{P}(G)), \cup, \Cup, \emptyset, \{\emptyset\} \rangle$. In AttributeSet TBAC, a credential $c \in \mathbb{C} = \mathcal{P}(G)$ is a subset of attributes the requester is assigned to. An access to $(t : k)$ is granted if the requester can fulfil all the requirements of at least one of the groups of attributes in k, formally: $f(c)(k) = \exists g \in k. g \subseteq c$. Table 2 recasts the example relation of Section 2 with the concrete structure of AttributeSet TBAC. For the tuple $(t_0 : k_0)$, the *s-tag* $k_0 = \{\{g_0, g_1\}, \{g_2\}\}$ means that any user who has attributes g_0 *and* g_1 *or* who has attribute g_2 has access to (a, b, c). Query result $q(r)$ of the Table 2 can be filtered using Bob's credentials $c = \{g_1, g_3\}$ as shown in the last column. Bob has access to (d, e) and (f, e) because his credentials cover one group of attributes of k_1 and another of k_2. Note that AttributeSet TBAC policies can be simplified by keeping only the minimal sets of attributes w.r.t. set inclusion. For instance, the element $\{\{g_0, g_1\}, \{g_0\}, \{g_1\}\} \in \mathcal{P}(\mathcal{P}(G))$ can be simplified to $\{\{g_0\}, \{g_1\}\}$. Such a property is used in formal proofs and can be seen as an optimization to keep the size of *s-tags* as small as possible. This simplification has been carried out on Table 2 for the sake of readability.

r	A	B	C	*s-tag*
t_0	a	b	c	$k_0 = \{\{g_0, g_1\}, \{g_2\}\}$
t_1	d	b	e	$k_1 = \{\{g_0\}, \{g_3\}\}$
t_2	f	g	e	$k_2 = \{\{g_1\}, \{g_2, g_3\}\}$

$q(r)$	A	C	*s-tag*	$f(c)(k)$
s_0	a	c	$\{\{g_0, g_1\}, \{g_2\}\}$	$\bot$
s_1	a	e	$\{\{g_0, g_1\}, \{g_0, g_2\}, \{g_2, g_3\}\}$	$\bot$
s_2	d	c	$\{\{g_0, g_1\}, \{g_0, g_2\}, \{g_2, g_3\}\}$	$\bot$
s_3	d	e	$\{\{g_0\}, \{g_3\}\}$	$\top$
s_4	f	e	$\{\{g_1\}, \{g_2, g_3\}\}$	$\top$

Table 2: AttributeSet TBAC example

3.1.3 Deadline TBAC

Among the rights covered by the 95/46/EC European directive on the processing of personal data, Article 6 (e) stipulates that personal data should be collected no longer than necessary. This right has been coined as *the right to oblivion*, which states that personal data will eventually vanish. The idea behind *Deadline* TBAC is to use *s-tags* as "best before dates": one should have access to a tuple if the current time is before

the deadline. In this setting, access rights are elements of $\mathbb{K} = \mathbb{N} \cup \{\infty\}$, where ∞ captures the lack of constraint. Credentials in $\mathbb{N} \setminus \{0\}$ denote the current time and are degenerated in the sense that they are not user-dependent. The decision function is defined by $f(c)(k) = c \leq k$ with $n \leq \infty$ for all $n \in \mathbb{N}$. The full semiring is finally $\langle \mathbb{N} \cup \{\infty\}, \max, \min, 0, \infty \rangle$.

3.2 Properties of the TBAC models family

A fundamental property of given TBAC instances is that for any given credentials the decision function behaves well in the following sense: 0 is the *deny all* policy, 1 is the *allow all* policy, one can read a tuple tagged with $X \oplus Y$ if he/she can read *either* X *or* Y, and one can read a tuple tagged with $X \otimes Y$ if he/she can read *both* X *and* Y. Informally, the decision functions of UserSet, AttributeSet and Deadline TBAC preserve the intuitive meaning of disjunction and conjunction of policies. This property is formally captured by stating that for all $c \in \mathbb{C}$, $f(c)$ is an *homomorphism* from $\langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle$ into the boolean semiring $\langle \mathbb{B}, \vee, \wedge, \bot, \top \rangle$. This is a key property of the approach: first it captures the semantics of access control, then it proves that TBAC fits into the $\mathcal{RA}^+$ framework, and finally, Corollary 1 ensures that TBAC can be seamlessly integrated into an R-DBMS in the sense that it is *independent* of the evaluation order of queries.

Proposition 1 *The curried decision function $f(c) : \mathbb{K} \to \mathbb{B}$ of UserSet (resp., AttributeSet and Deadline) is a homomorphism for all $c \in \mathbb{C}$.*

The Fundamental Theorem of $\mathcal{RA}^+$ [13, Theorem 3.3] shows that the semiring structure works nicely with SPJRU operations: homomorphism of semirings and $\mathcal{RA}^+$ queries commutes. Applied to TBAC, the Fundamental Theorem ensures that the decision function can be applied either *before or after the evaluation of a query without loss*. This guarantees that algebraic query optimization, query rewriting and other techniques that rely on equivalent algebraic expressions can still be used transparently in the presence of TBAC.

When a query is executed, its result is filtered through the decision function $f(c)$. The filtered result of $q(r)$ is formally defined by the following function eval_1 that selects the subset of authorized tuples *after* the evaluation of $q(r)$: $\text{eval}_1(r)(q, c) = \{(t : k) \mid (t : k) \in q(r) \wedge f(c)(k)\}$. Alternatively, it is possible to apply the decision function *before* the evaluation of $q(r)$, that is, we define another evaluation function eval_2: $\text{eval}_2(r)(q, c) = q(r')$ with $r' = \{(t : k) \mid (t : k) \in r \wedge f(c)(k)\}$. Corollary 1 is obtained by applying the Fundamental Theorem of $\mathcal{RA}^+$ to Proposition 1.

Corollary 1 *The filtering function of TBAC can be applied* before *or* after *the evaluation of a query without loss, that is* $\text{eval}_1(r)(q, c) = \text{eval}_2(r)(q, c)$.

3.3 Integrating heterogeneous policies

We show now that we can exploit the structure of semirings to combine heterogeneous instances of TBAC together into larger ones, mimicking an important property of lattice-based access control models.

Let $\langle \mathbb{K}_i, \oplus_i, \otimes_i, 0_i, 1_i \rangle$ be a finitely I-indexed family of semirings and let $\mathbb{K} = \mathbb{K}_0 \times \ldots \times \mathbb{K}_n$ be the Cartesian product of their underlying sets. We denote by $\pi_i : \mathbb{K} \to \mathbb{K}_i$ the ith projection of the product that maps a tuple to its ith component. The set $\mathbb{K}$ equipped with *componentwise* addition $\langle k_0, \ldots, k_n \rangle \oplus \langle k'_0, \ldots, k'_n \rangle = \langle k_0 \oplus_0 k'_0, \ldots, k_n \oplus_n k'_n \rangle$ and componentwise multiplication is itself a semiring called the *product semiring* [11]. For instance, we can integrate UserSet, AttributeSet and Deadline TBAC structures altogether by building the following product structure $\mathcal{P}(U) \times \mathcal{P}(\mathcal{P}(G)) \times (\mathbb{N} \cup \{\infty\})$ with addition of policies defined by $(x, y, z) \oplus (x', y', z') = (x \cup x', y \cup y', \max(z, z'))$ and multiplication defined by $(x, y, z) \otimes (x', y', z') = (x \cap x', y \Cup y', \min(z, z'))$.

The decision function $f : \mathbb{C} \times \mathbb{K} \to \mathbb{B}$ can be defined from the f_is using either a *Deny Takes Precedence* (DTP) conflict resolution rule defined by $f(c)(k) = \forall i \in I.f_i(c)(\pi_i(k))$ or a *Permit Takes Precedence* (PTP) conflict resolution rule defined by $f(c)(k) = \exists i \in I.f_i(c)(\pi_i(k))$. Following the example mixing UserSet,

AttributeSet and Deadline, we obtain the following definition of the combined function in the PTP case: $f(u, gs, n)(X, Y, Z) \Leftrightarrow u \in X \wedge \exists g \in Y.g \subseteq gs \wedge n \leq Z$.

However, if we want to combine two *existing* relations already tagged with different structures into a large one we need a method to consider each atomic policy as a member of the large one. In other words, we need a way to canonically inject the existing policies $\mathbb{K}_i$ into the product structure $\mathbb{K}$. This can be done using the following family of functions: $\iota_i : \mathbb{K}_i \to \mathbb{K}$, one for each $i \in I$, defined by combining a given *s-tag* with the "authorize all" policies for other components: $\iota_i(k) = (1_0, \ldots, 1_{i-1}, k, 1_{i+1}, \ldots, 1_n)$. For example, the injection $\iota_0 : \mathcal{P}(U) \to \mathbb{K}$ is defined by $\iota_0(x) = (x, \{\emptyset\}, \{\infty\})$.

Proposition 2 *By combining semirings and decision functions as defined above, the product structure is a semiring and the decision function is an homomorphism, with both DTP and PTP conflict resolution rules.*

Interestingly, Proposition 2 ensures that one does not need to always use the same conflict resolution rule. For instance, by integrating two semirings at a time: first $\mathcal{P}(U)$ and $\mathcal{P}(\mathcal{P}(G))$ and then $(\mathbb{N} \cup \{\infty\})$, one can select DTP for the first pair and then PTP.

4 Path-based dissemination

In the previous section we have defined and illustrated the TBAC framework with three standard access control structures. In this section we present a new structure for *s-tags*, called *Path* TBAC. The motivation driving Path TBAC is to introduce the concept of consumption in *s-tags* to be able to control the depth of a data dissemination. The security semantics is not only to filter the tuples pertaining to a query result but also to *modify* the *s-tags* obtained during query evaluation. We obtain a resource-limited information flow model where security tags are consumed when tuples are transferred from one PDS to another.

Let the set U denote the set of PDS identifiers. The set $U^\star$ of finite strings over the alphabet U can be endowed with the *prefix partial order* by defining $x \sqsubseteq y \Leftrightarrow \exists z.(x :: z) = y$ with $(x :: y)$ being the concatenation of two words. This partial order is equipped with a *greatest lower bound* operator on pairs of strings which is the *longest common prefix* written $\mathrm{lcp}(x, y)$. An *s-tag* is a *set* of authorized paths with label taken from U. To capture the lack of constraints on data dissemination, we add an extra element $*$, thus the set of all Path TBAC policies becomes $\mathbb{K} = \mathcal{P}(U^\star) \cup \{*\}$. For instance, the *s-tag* $k_0 = \{(B, D)\}$ captures the fact that the tuple t_0 can go first to Bob (B) and *then* to Denise (D). The $\oplus$ policy combinator is the union of sets of paths $X \cup Y$ with $X \oplus * = *$. The empty set plays the role of neutral element for $\cup$. The $\otimes$ combinator is defined similarly to the $\Cup$ operator by $X \otimes Y = \{\mathrm{lcp}(x, y) \mid x \in X \wedge y \in Y\}$. In order to control the depth of the dissemination, the decision function $f(c)$ is defined according to the locality of *who* is requesting the tuple. If the destination PDS is the current PDS, then $f(c)$ returns $\top$. If the destination is remote, $f(c)$ returns $\top$ only if *at least one of the paths* of the *s-tag* begins with the remote destination, that is $f(c)(k) = \exists ps.(c :: ps) \in k$. The following proposition ensures that this structure is indeed a semiring and that $f(c)$ is another semiring homomorphism.

Proposition 3 $\langle \mathbb{K} = \mathcal{P}(U^\star) \cup \{*\}, \oplus, \otimes, \emptyset, * \rangle$ *is a commutative semiring and* $f(c)$ *is an homomorphism.*

When a tuple leaves a PDS its *s-tag* must be updated to reflect this hop. For each tuple transferred to c, each path of the annotation is modified by popping c from the beginning of each path. The update function $\mathrm{upd} : \mathbb{C} \times \mathbb{K} \to \mathbb{K}$ is formally defined by: $\mathrm{upd}(c)(k) = \{p \mid (c :: p) \in k\}$. The evaluation function is now upgraded to the following definition where $q(r)$ is filtered and the *s-tags* modified: $\mathrm{eval}(r)(q, c) = \{(t : \mathrm{upd}(c)(k)) \mid (t : k) \in q(r) \wedge f(c)(k) = \top\}$.

Table 3 recasts the running example with Path *s-tags*. The transfer scenario is made of two steps. First, Bob queries Alice's PDS to obtain $q(r)$ and store it in his own PDS. Then, Charlie asks Bob for the whole content of $q(r)$ and obtains $q'(r)$ which is a strict subset of $q(r)$. At this point Charlie holds the tuple s'_3 but cannot transfer it anymore, however, s'_4 can be freely distributed. If Charlie had asked Alice directly, the disclosure would have been limited to $\{(f, e) : \{*\}\}$ because Charlie has only access to t_2.

This ability to define fine-grained control policies with different results according to the previous step is a key feature of Path TBAC. In the example, Charlie receives *less* information by querying Alice, who is

r	A	B	C	*s-tag*
t_0	a	b	c	$k_0 = \{(B, D)\}$
t_1	d	b	e	$k_1 = \{(B, C), (B, D)\}$
t_2	f	g	e	$k_2 = \{*\}$

Alice's relation r

$\Rightarrow$

$q(r)$	A	C	*s-tag*
s_0	a	c	$\{(D)\}$
s_1	a	e	$\{(D)\}$
s_2	d	c	$\{(D)\}$
s_3	d	e	$\{(C), (D)\}$
s_4	f	e	$\{*\}$

Bob's result $q(r)$

$\Rightarrow$

$q'(r)$	A	C	*s-tag*
s'_3	d	e	$\{()\}$
s'_4	f	e	$\{*\}$

Charlie's result $q'(r)$

Table 3: Path TBAC example

supposed to be the originator of data, than by querying Bob. Such a scenario is relevant in many contexts and is coined as *intransitive information flow*. For instance, Bob may be a trusted party that ensures encryption, or Bob may be Charlie's boss who has to control the data sent to his subordinates. In this last example, Alice trusts Bob to select the right subset of data Charlie is authorized to.

5 Related Work

According to the classification used in the monograph entitled *Access Control for Databases* [5, Section 5], the TBAC control model can be categorized as a tuple-labeled access control model. View-based access control is the most common access control mechanism in R-DBMS but lacks flexibility and scalability because of the necessary proliferation of views. A more flexible and transparent approach is to try to rewrite the user's query using authorized views and to grant access if the rewriting succeeds [20]. Even so, privileges are still not attached to the data items themselves but to their containers.

Whereas MAC and related information flow models provide strong guarantees [21, 9], we advocate that the context of personal information management and PDSs presented in Section 1 would benefit from a softer user-based approach. In TBAC, the classification of a tuple is not derived from the user's credentials but is user-specified at insertion time. TBAC relates to the Hippocratic database paradigm where owners specify to where data can flow. According to the classification of LeFevre *et al.* on Hippocratic databases [14], TBAC follows the *query semantics disclosure model*: a tuple is discarded or not, without partial releases using `NULL` for forbidden attributes.

The TBAC dissemination function is close to the concepts of originator control, sticky policies and dissemination control. We give hereafter a brief overview of some related work. Park and Sandhu [19] study the combination of originator control with usage control. Thomas and Sandhu [24] provide an overview of dissemination control characteristics. Sandhu *et al.* [22] propose a *Policy, Enforcement, Implementation* model for secure information sharing, targeting TPM-enabled architectures (trusted software), which share similarities with PDSs. Bandhakavi *et al.* [4] define a logic framework to specify release control policies. Sticky release policies are used to control future dissemination of the data, and also of the aggregated data. Finally, Cerbo *et al.* [8] propose to use sticky policies to integrate usage control conditions in mobile devices. The main purpose of the dissemination function in TBAC is to tackle the problems of data combinations of relational data for which it is tailored.

Provenance-Based Access Control have been coined in 2011 by Cadenhead *et al.* [7] and then by Park *et al.* [18] with the following manifesto: "*we strongly feel that access control systems built upon provenance data by fully utilizing its unique characteristics will provide a foundation for new access control mechanisms that are highly capable of supporting features that were not easily achievable with traditional access control solutions*". The initial proposal by Cadenhead *et al.* turned its focus to the RDF triple-oriented data model

and the applicability of access control technologies, whereas the second one by Park *et al.* sticks to the TBAC rationale. However, this second contribution provides a very wide vision on such models and does not define any specific model.

Provenance-based access control has received attention from the semantic web community. Lopes *et al.* [16] focus on the provenance machinery for the RDF-S paradigm. They use attribute-based policies with negation. Papakonstantinou *et al.* [17] provide a similar system with more detailed experiments. Whereas these papers specify a unique concrete annotation domain, TBAC provides several different ones with a mechanism to integrate them. Moreover, we broaden the scope of applicability with the Path TBAC model that controls dissemination in a distributed environment.

6 Discussion

In this section we discuss different issues related to the implementation of TBAC. The first issue is the formal language of *s-tags*. Elements of UserSet, AttributeSet and Deadline TBAC all have a formal representation and a natural implementation. In general, it is also the case that *any* semiring has a syntactical representation because the category of semirings has an initial element, namely $\mathbb{N}[X]$: the set of formal polynomials with integer coefficients freely generated from a set X of variables [11]. As $\mathbb{N}[X]$ can be easily implemented using a recursive datatype, it is an adequate generic formal representation of *s-tags*. For instance, AttributeSet is obtained from $\mathbb{N}[G]$ by dropping coefficients and exponents.

Regarding user-friendliness, one may not assume that end-users will express their security policies by writing formal polynomials. The idea is rather to provide high-level, possibly graphical, languages to users and then to compile these expressions into concrete *s-tags*. As an example, one can introduce a set R of roles with a GUI to graphically define user-role assignments ($UR \subseteq U \times R$) and tuple-role assignments ($RT \subseteq R \times T$) to help end-users writing (possibly large) UserSet expressions. The *s-tag* k associated with a tuple t is computed by $k = \{u | \exists r \in R.(u, r) \in UR \wedge (r, t) \in TR\}$. As ongoing work, we are studying high-level statements such as "I authorize `any member` of `this group` to access `this class of data` as long as the data are `served by` x and that `I have access to my own data`" to be translated into Path TBAC expressions.

Enforcement is a key issue for access control models. The use of cryptographic systems can ensure the security of data while transferred between PDSs. Whereas PDSs are assumed to be trusted by all users in the system, it is not the case for communication services. A solution is to encrypt tuples according to their *s-tags* by using related work on the *Ciphertext-Policy Attribute-Based Encryption* (CP-ABE) [6]. In CP-ABE, attributes are used to describe users' credentials, and a party encrypting data determines a policy for who can decrypt. Once applied to TBAC, the idea amounts to letting PDSs encrypt the tuples they emit according to the *s-tags*. CP-ABE deals with ciphertext-policies that can represent positive boolean formulae. As AttributeSet policies are isomorphic to positive boolean formulae, CP-ABE can be used for it. Extending CP-ABE to deal with richer semirings and ultimately $\mathbb{N}[X]$ is left open.

7 Conclusion

Motivated by personal information management in a federation of personal databases, this paper has introduced a tuple-grained control model for relational data named TBAC. The model is grounded in the algebraic foundations of the provenance data model, guaranteeing transparent integration of MAC into classical query evaluation engines of R-DBMSs. Moreover, we showed that the framework can be used to provide fine-grained user-based dissemination control using Path TBAC.

We have developed a proof-of-concept implementation of TBAC using the Perm prototype: a R-DBMS that handles provenance [10]. Perm is used as the database tier of an n-tiers application, where the TBAC filtering process is integrated into the application server that acts as a proxy between users and the DB tiers. An ongoing funded research project is to integrate TBAC into an existing R-DBMS, small enough to

fit into a secure smart card. We have also developed a second proof-of-concept where PDSs are used control collection and dissemination of personal data in a smart building scenario [15].

Besides experiments, an ongoing research direction is to provide a completely algebraic definition of the TBAC models family. Actually, the filtering functions are not defined using the $\otimes$ and $\oplus$ operators but directly on the concrete structures of $\mathbb{C}$ and $\mathbb{K}$. For instance, the condition $f(c)(k) \Leftrightarrow c \in k$ for UserSet is equivalent to $k \cup \{c\} = k$ and $f(c)(k) \Leftrightarrow c \leq k$ for Deadline is equivalent to $\max(c, k) = k$. These evidences suggest that the class of interesting semirings for security applications is the class of *canonically ordered semirings*, where the relation $a \leq b$ given by $\exists c.a \oplus c = b$ is a partial order [11].

Our last research avenue is to extend the positive fragment of the provenance framework to richer languages, including aggregation or difference. Theoretical results that extend the $\mathcal{RA}^+$ framework have been obtained [2], however, the algebraic structures are more *ad-hoc* than semirings and their applicability to access-control purposes is left open. Being able to provide a systematic definition of the decision function without reference to the internal structure of $\mathbb{C}$ and $\mathbb{K}$ may lead to a more generic framework, suitable for these extensions.

Acknowledgments

The authors gratefully thank the anonymous referees for their constructive suggestions and thorough reviews. This research was partially funded by the French ANR KISS project under grant No. ANR-11-INSE-0005.

References

[1] Agrawal, R., Kiernan, J., Srikant, R., and Xu, Y. Hippocratic databases. In *VLDB* (2002), pp. 143–154.

[2] Amsterdamer, Y., Deutch, D., and Tannen, V. Provenance for aggregate queries. In *PODS* (2011), pp. 153–164.

[3] Anciaux, N., Bouganim, L., Pucheral, P., Guo, Y., Le Folgoc, L., and Yin, S. Milo-db: a personal, secure and portable database machine. *Distributed and Parallel Databases* (2013), 1–27.

[4] Bandhakavi, S., Zhang, C. C., and Winslett, M. Super-sticky and declassifiable release policies for flexible information dissemination control. In *WPES* (2006), ACM.

[5] Bertino, E., Ghinita, G., and Kamra, A. Access control for databases: Concepts and systems. *Found. Trends databases 3* (Jan. 2011), 1–148.

[6] Bethencourt, J., Sahai, A., and Waters, B. Ciphertext-policy attribute-based encryption. In *Security and Privacy* (2007), pp. 321–334.

[7] Cadenhead, T., Khadilkar, V., Kantarcioglu, M., and Thuraisingham, B. M. A language for provenance access control. In *CODASPY* (2011), ACM, pp. 133–144.

[8] Cerbo, F. D., Trabelsi, S., Steingruber, T., Dodero, G., and Bezzi, M. Sticky policies for mobile devices. In *SACMAT* (2013), ACM.

[9] Cuppens, F., and Gabillon, A. Logical foundations of multilevel databases. *Data & Knowledge Engineering 29*, 3 (1999), 259–291.

[10] Glavic, B., and Alonso, G. Perm: Processing provenance and data on the same data model through query rewriting. In *ICDE* (2009), pp. 174–185.

[11] Gondran, M., and Minoux, M. *Graphs, Dioids and Semirings: New Models and Algorithms*. 2008.

[12] Karjoth, G., Schunter, M., and Waidner, M. Platform for enterprise privacy practices: Privacy-enabled management of customer data. In *Privacy Enhancing Technologies* (2002), pp. 69–84.

[13] Karvounarakis, G., and Green, T. J. Semiring-annotated data: queries and provenance? *SIGMOD Rec. 41*, 3 (Oct. 2012), 5–14.

[14] LeFevre, K., Agrawal, R., Ercegovac, V., Ramakrishnan, R., Xu, Y., and DeWitt, D. Limiting disclosure in hippocratic databases. In *VLDB* (2004).

[15] Lesueur, F., Surdu, S., Thion, R., Gripay, Y., and Talbi, M. Palpable privacy through declarative information flows trackingfor smart buildings. In *ARES Conference* (2014), pp. 1–6.

[16] Lopes, N., Kirrane, S., Zimmermann, A., Polleres, A., and Mileo, A. A logic programming approach for access control over rdf. In *ICLP* (2012), pp. 381–392.

[17] Papakonstantinou, V., Michou, M., Fundulaki, I., Flouris, G., and Antoniou, G. Access control for rdf graphs using abstract models. In *ACM SACMAT* (2012), pp. 103–112.

[18] Park, J., Nguyen, D., and Sandhu, R. S. A provenance-based access control model. In *PST* (2012), IEEE, pp. 137–144.

[19] Park, J., and Sandhu, R. Originator control in usage control. In *POLICY* (2002), pp. 60–66.

[20] Rizvi, S., Mendelzon, A., Sudarshan, S., and Roy, P. Extending query rewriting techniques for fine-grained access control. In *ACM SIGMOD* (2004).

[21] Rjaibi, W., and Bird, P. A multi-purpose implementation of mandatory access control in relational database management systems. In *VLDB* (2004), pp. 1010–1020.

[22] Sandhu, R., Ranganathan, K., and Zhang, X. Secure information sharing enabled by Trusted Computing and PEI models. In *ASIACCS* (2006).

[23] Sandhu, R. S. Lattice-based access control models. *Computer 26*, 11 (1993), 9–19.

[24] Thomas, R., and Sandhu, R. Towards a multi-dimensional characterization of dissemination control. In *POLICY* (2004), pp. 197–200.