11
Toward a MAC policy framework

Xiaolei Qian
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
qian@csl.sri.com

Teresa F. Lunt
ARPA/ITO
3701 North Fairfax Drive
Arlington, VA 22203
tlunt@arpa.mil

Abstract

We propose a formal policy framework of MAC policies in multilevel relational databases. We identify the important components of such policies and their desirable properties. The framework provides a basis for systematically specifying such policies and characterizing their potential mismatches. Based on the framework, we compare and unify the MAC policies and policy components that are proposed in the literature or imposed in existing systems. Our framework could be used to capture and resolve MAC policy mismatches in the trusted interoperation of heterogeneous multilevel relational databases.

Keywords

Heterogeneity, interoperation, mandatory access control, multilevel security, relational database, security policy

1 INTRODUCTION

As more multilevel databases are built and connected through computer networks, a wide variety of secure data sources will become accessible. A big challenge presented by this technology is the trusted interoperation of multilevel databases containing data with mismatched security policies. Providing trusted interoperation of multilevel databases not only makes it possible to reliably share data in isolated military and civilian databases, but also increases users' confidence and willingness in such sharing.

As a prerequisite to the trusted interoperation of multilevel databases containing data with mismatched security policies, the security policies of component databases, as well as the potential mismatches between them, have to be precisely characterized. The security policy of a multilevel database captures the security requirements of an application that govern the access and manipulation of data. These requirements can be very complicated, ranging from high-level specifications such as the type of access control (mandatory or discretionary) or the kind of model (noninterference or Bell-LaPadula), to designer's belief or preferences such as whether polyinstantiation is allowed, to low-level implementation decisions such as the number of levels and categories allowed in a lattice. A formal policy framework is needed within which security policies could be characterized and compared

D. L. Spooner et al. (eds.), *Database Security IX*

(Hosmer, 1990). As a first step in this direction, we propose a formal policy framework for an important component of security policies: the mandatory access control (MAC) policy.

It has been widely accepted that a MAC policy consists of four components: a set of subjects, a set of objects, a lattice, and a mapping that associates levels in the lattice to subjects and objects (Landwehr, 1981). This works fine with multilevel operating systems, because objects such as files do not carry semantics. For multilevel databases where data carry semantics, the same mapping of levels to objects such as elements in tuples could have completely different meanings. For example, consider a relation SMD(Starship, MId, Destination). A secret label on element Rigel of tuple (Enterprise, 101, Rigel) in SMD could mean that the fact "Enterprise is going to Rigel" is secret, or the fact "some starships are going to Rigel" is secret, or even the word "Rigel" is secret. This confusion suggests that something critical is missing with the traditional formulation of MAC policies in multilevel databases, namely the semantics of object labels. This problem is crucial in the trusted interoperation of multilevel databases. For example, if the secret label on Rigel means that the fact "some starships are going to Rigel" is secret in database A, and means that the word "Rigel" is secret in database B, then unclassified subjects could query all destinations in database A and obtain "Rigel" through interoperation with database B. The canonical MAC policy for federated databases proposed in (Pernul, 1992) does not solve this problem.

The formulation of a MAC policy in a multilevel database often includes some constraint policies, such as the labeling policy of Seaview (Lunt, 1989) and the classification constraints of LDV (Haigh, 1991). Constraints are the most important means of specifying data semantics. However, the MAC policies in existing multilevel databases provide neither a precise definition of constraint validity nor an efficient mechanism of constraint enforcement. In fact, it has been argued (Burns, 1990), (Meadows, 1988) that integrity enforcement is in fundamental conflict with secrecy enforcement: no multilevel databases could simultaneously satisfy both integrity and secrecy requirements.

An important characteristic of MAC policies is the upward information flow in the lattice, which indicates the believability of low data at high levels. For multilevel operating systems where objects do not carry semantics, low data are always believed at high. For multilevel databases where data carry semantics expressed by constraints however, low data could contradict high data. For example, if we require that high SMD tuples have unique MId elements and (Enterprise, 101, Rigel) is a high tuple in SMD, then the low tuple (Enterprise, 102, Rigel) in SMD could not be believed at high. This problem suggests that the formulation of MAC policies in multilevel databases should provide means to constrain upward information flow.

Constraints also bring about the danger of inference channels. Inference channels could be obtained by observing the behavior of a database in enforcing the constraints.* In other words, the result of a low update could violate some constraints when combined with high data, but prohibiting the low update would enable the low subject to infer the existence of relevant high data. For example, consider another relation MT(MissionId, Type). If we require that every high MId element in SMD refers to a high or low MissionId element in

*Because inference channels involve system behavior, in many cases the mechanisms could also be used as covert channels between cooperating malicious high and low subjects. However, we concern ourselves here only with undesired inferences through such mechanisms, which do not require either a high "sender" or that the low "receiver" be malicious.

MT, then prohibiting the deletion of a low MissionId element referred to by a high MId element would enable low users to infer the existence of the high MId element. Thus the formulation of MAC policies in multilevel databases should provide means to detect and remove such inference channels.

Existing research on security policies has focused on using first-order logic to specify, analyze, and enforce the security requirements of an application (see for example (Michael, 1993), (Morris, 1992), (Sibley, 1992), (Steinke, 1993)). Security policies are formulated in terms of a set of subjects, a set of objects, and a set of rules governing the various modes of access of subjects to objects. There are several problems associated with applying this approach to MAC policies. The objects are usually the physical containers of data rather than the semantics of data, and are often too coarse in granularity (e.g., buildings in (Michael, 1993) and files in (Morris, 1992)). Moreover, first-order logic is not expressive enough to capture the upward information flow in the lattice, or the dynamic behavior of a database in constraint enforcement. In addition, no methods are available to translate the logical statements in security policy specifications to the labels and constraints in multilevel databases.

We propose a formal policy framework for MAC policies in multilevel relational databases. We identify the important components of MAC policies and their desirable properties. The framework abstracts away the superficial syntactic difference and makes precise the hidden semantic difference in MAC policy specifications. It provides a basis for systematically specifying MAC policies and characterizing their potential mismatches. Based on the framework, we are able to compare and unify the MAC policies and policy components that are proposed in the literature or imposed in existing systems.

The rest of the paper is organized as follows. In Section 2, we describe our framework and identify the components of MAC policies. Besides the standard components of such policies, we identify three new components—an interpretation policy, a view policy, and an update policy—as essential for multilevel relational databases. In Sections 3 through 5, we discuss the three most important new components of our policy framework, and specify and unify MAC policy components that are proposed in the literature or imposed in existing systems. Finally, Section 6 offers concluding remarks and a brief discussion of future work.

2 A POLICY FRAMEWORK

2.1 A Model-Theoretic Formulation

We distinguish between the *actual world* and a *perceived world*. A perceived world is a view of the actual world as perceived by a group of users. Information in a perceived world is the knowledge (of a group of users) of the truth value of a statement about the actual world (Nicolas, 1978), which could be either an elementary fact such as "Enterprise is on mission #101 to Rigel" or a general law such as "starships have unique missions".

A (single-level) *relational database* captures information in a perceived world—the view of the actual world as perceived by users of the database. Elementary facts are represented as tuples in relations, and general laws are represented as integrity constraints. For example, the elementary fact "Enterprise is on mission #101 to Rigel" could be represented by

the tuple (Enterprise, 101, Rigel) in relation SMD, and the general law "starships have unique missions" is represented by a functional dependency SMD: Starship → MId.

A standard model-theoretic formulation of a relational database is to interpret relation names as predicates, integrity constraints as axioms in a first-order theory, and relations as forming a first-order structure of the theory (Nicolas, 1978). A database is valid if the structure is a model of the theory. For example, the tuple (Enterprise, 101, Rigel) in relation SMD is interpreted as a tuple in the assignment to predicate SMD, and the functional dependency SMD: Starship → MId is interpreted as the axiom (variables are universally quantified)

$$\mathrm{SMD}(x, y_1, z_1) \wedge \mathrm{SMD}(x, y_2, z_2) \rightarrow y_1 = y_2.$$

A *multilevel perceived world* is a family of perceived worlds organized into a lattice. A perceived world at a level in the lattice is the view of the actual world as perceived by subjects at that level.† Information in a multilevel perceived world is either information in the perceived worlds or knowledge of relationships between the perceived worlds. The former could be either a classified elementary fact such as "it is top-secret that Enterprise is on mission #101 to Rigel", or a classified general law such as "confidential starships have unique missions". The latter could be a general law on classification such as "starships classified at all levels have unique missions".

A *multilevel relational database* captures information in a multilevel perceived world. It consists of a relational database, whose integrity constraints are called *view constraints*, together with a *labeling function* κ and a set of *labeling constraints*. The labeling function maps every object in the database—relation, attribute, tuple, element in a tuple, view constraint, etc.—to a (possibly empty) set of levels in the lattice. The database and the labeling function together represent the family of perceived worlds, and the labeling constraints represent the general laws on classification. For example, the tuple (Enterprise, 101, Rigel) mapped to `ts` by κ represents the classified elementary fact "it is top-secret that Enterprise is on mission #101 to Rigel". As a view constraint, the functional dependency SMD: Starship → MId mapped to `c` by κ represents the classified general law "confidential starships have unique missions". As a labeling constraint, the functional dependency SMD: Starship → MId represents the general law on classification "starships classified at all levels have unique missions".

The above observation suggests a model-theoretic formulation of multilevel relational databases as follows. A *multilevel theory* is a triplet $(\mathcal{L}, \{T^l\}_{l \in L}, \mathcal{A})$:

1. $\mathcal{L} = (L, \preceq)$ is a lattice where L is a set of *levels* and $\preceq$ is the *dominance relation*,
2. $\{T^l\}_{l \in L}$ is a family of first-order theories—one for every level in L, each of which representing the view constraints that are mapped to a particular level, and
3. $\mathcal{A}$ is a collection of axioms representing the labeling constraints.

We use $\prec$ to denote the strict dominance subrelation, and $\preceq^*$ to denote the transitive closure of $\preceq$. A *multilevel structure* of the multilevel theory is a family of first-order structures $\{M^l\}_{l \in L}$ where M^l is a structure of theory T^l.

†These perceived worlds differ, because the actual world allowed to be known and understood differs for subjects at different levels.

For example, the tuple (Enterprise, 101, Rigel) in relation SMD mapped to `ts` by κ is interpreted as a tuple in the assignment to predicate $\mathrm{SMD}^{\mathtt{ts}}$ in structure $M^{\mathtt{ts}}$, the view constraint SMD: Starship $\rightarrow$ MId mapped to `c` by κ is interpreted as the axiom

$$(\forall x, y_1, y_2, z_1, z_2)(\mathrm{SMD}^{\mathtt{c}}(x, y_1, z_1) \wedge \mathrm{SMD}^{\mathtt{c}}(x, y_2, z_2) \rightarrow y_1 = y_2)$$

in theory $T^{\mathtt{c}}$, and the labeling constraint SMD: Starship $\rightarrow$ MId is interpreted as the axiom

$$(\forall l_1, l_2 \in \mathcal{L})(\forall x, y_1, y_2, z_1, z_2)(\mathrm{SMD}^{l_1}(x, y_1, z_1) \wedge \mathrm{SMD}^{l_2}(x, y_2, z_2) \rightarrow y_1 = y_2)$$

in $\mathcal{A}$.

2.2 MAC Policy

We restrict ourselves to multilevel relational databases whose MAC policies have the simple security property and the $*$-property of the Bell-LaPadula model (Landwehr, 1981), which ensure that information does not flow downward in the lattice.

- **The Simple Security Property.** A subject is allowed a read access to an object only if the former's clearance level is identical to or higher than the latter's classification level in the lattice.
- **The $*$-Property.** A subject is allowed a write access to an object only if the former's clearance level is identical to or lower than the latter's classification level in the lattice.

Our formulation of a MAC policy in a multilevel relational database has eight components:

1. a lattice,
2. a set of subjects,
3. a set of objects,
4. a mapping of subjects and objects to levels in the lattice,
5. a set of labeling constraints,
6. an interpretation policy,
7. a view policy, and
8. an update policy.

The first five components together correspond to the traditional formulation of MAC policies in multilevel relational databases.

An interpretation policy maps a multilevel relational database to a multilevel theory and a multilevel structure of the multilevel theory. Through this policy, the superficial syntactic difference in object labels is abstracted away, and the semantic difference hidden in object labels is made precise. As a consequence, the interpretation policy makes it possible to compare the semantics of multiple MAC policies.

A view policy is a specification of the upward information flow—believability of low data at high levels. For every level, view constraints at that level are enforced on data believable at that level.

An update policy consists of a set of updates and a specification of the enforcement of labeling constraints on visible data in performing the updates, such that inference channels are eliminated in the enforcement.

In the rest of this paper, we investigate the last three components of our policy framework, using examples from multilevel relational databases based on the lattice in Figure 1.

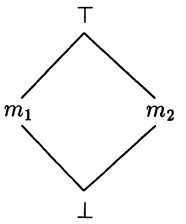

Figure 1 A Lattice

3 INTERPRETATION POLICY

An *interpretation policy* maps a multilevel schema to a multilevel theory and a multilevel database to a multilevel structure of the multilevel theory. Through this mapping, the superficial syntactic difference in object labels is abstracted away, and the semantic difference hidden in object labels is made precise. As a consequence, the interpretation policy makes it possible to compare the semantics of multiple MAC policies.

Here, we investigate the interpretation policy for the most common multilevel databases, namely multilevel databases with tuple-level labeling, where objects are tuples. In (Qian, 1993), we developed an interpretation policy for multilevel databases with element-level labeling.

3.1 Multilevel Relational Model

Let U be a finite set of *attributes*. If X, Y are subsets of U, then XY denotes the union of X, Y. If $A \in U$, then XA denotes $X\{A\}$. A *relation scheme* (in Boyce-Codd Normal Form) $R[X, K]$ is a set of attributes $X \subseteq U$ named R with nonempty *primary key* $K \subseteq X$. A *database schema* is a pair $(\mathcal{R}, \mathcal{C})$, where $\mathcal{R} = \{R_i[X_i, K_i]\}_{1 \leq i \leq n}$ is a family of relation schemes and $\mathcal{C}$ is a set of *key-based referential dependencies*:

- Every referential dependency in $\mathcal{C}$ has the form $R_i[Y] \hookrightarrow R_j$, where $1 \leq i, j \leq n$, $Y \subseteq K_i$ or $Y \subseteq X - K_i$, and $|Y| = |K_j|$. Y is a *foreign key* in relation scheme R_i to relation scheme R_j.
- Distinct foreign keys in the same relation scheme are disjoint. In other words, $Y = Z$ or $Y \cap Z = \emptyset$ for $1 \leq i, j, k \leq n$ and $R_i[Y] \hookrightarrow R_j, R_i[Z] \hookrightarrow R_k$ in $\mathcal{C}$.

For relation scheme $R_i[X_i, K_i]$ in $\mathcal{R}$ and attribute $A \in X_i$, A is a *nonkey* attribute if $A \notin K_i$ and $A \notin Y$ for any foreign key Y in R_i. Figure 2 shows a schema with two relation schemes SMD and MT, where boxes represent relation schemes, attributes to the left of double lines form primary keys, and arrows between boxes represent referential dependencies.

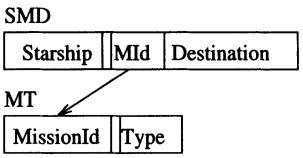

Figure 2 A Schema

Let $\mathcal{D}$ be a (possibly infinite) set of values. A *tuple* over attributes X is a partial mapping $t[X]: X \mapsto \mathcal{D}$ that assigns values from $\mathcal{D}$ to attributes in X. For attribute $A \in X$, $t[A]$ denotes the value assigned to A by $t[X]$, and $t[A] = \perp$ denotes that $t[A]$ is undefined. For attributes $Y \subseteq X$, $t[Y]$ denotes the partial mapping whose domain is restricted to attributes in Y. For tuple t over X, $t[X] = \perp$ denotes that t is empty: $t[A] = \perp$ for all attributes $A \in X$; and $t[X] \neq \perp$ denotes that t is total: $t[A] \neq \perp$ for all attributes $A \in X$.

A *relation* r over relation scheme $R[X, K]$ is a set of tuples over X. For attributes $Y \subseteq X$, $r[Y]$ denotes the set of tuples $t[Y]$ where $t \in r$. Relation r satisfies the *key integrity property* if:

- for every tuple $t \in r$, $t[K] \neq \perp$, and
- for every pair of tuples $t, t' \in r$, $t[K] = t'[K]$ implies $t = t'$.

In other words, tuples with the same primary key value are identical.

A *database* b over schema $(\mathcal{R}, \mathcal{C})$, where $\mathcal{R} = \{R_i[X_i, K_i]\}_{1 \leq i \leq n}$, is a family of relations $\{r_i\}_{1 \leq i \leq n}$, where r_i is a relation over $R_i[X_i, K_i]$. It satisfies the *referential integrity property* for referential dependency $R_i[Y] \hookrightarrow R_j$ in $\mathcal{C}$ and tuple $t \in r_i$ if:

- either $t[Y] = \perp$ or $t[Y] \neq \perp$, and
- if $t[Y] \neq \perp$ then there is a tuple $t' \in r_j$ such that $t[Y] = t'[K_j]$.

In other words, every non-null foreign key refers to an existing primary key. $\mathcal{D}$ is the *universe* of b. Below is a database over the schema of Figure 2 that satisfies the key and referential integrity properties.

Starship	Mission	Destination
Enterprise	101	Rigel
Voyager	102	Talos
Discovery	103	Rigel

MissionId	Type
101	spy
102	explore
103	mine

A *multilevel relation scheme* is a pair $(R[X,K],\mathcal{L})$, where $R[X,K]$ is a relation scheme and $\mathcal{L}$ is a lattice. A *multilevel database schema* is a pair $(\mathcal{B},\mathcal{L})$, where $\mathcal{B}$ is a schema and $\mathcal{L}$ is a lattice.

Let $(\mathcal{B},\mathcal{L})$ be a multilevel schema, where $\mathcal{B} = (\mathcal{R},\mathcal{C})$, $\mathcal{R} = \{R_i[X_i,K_i]\}_{1\le i\le n}$, and $\mathcal{L} = (L,\preceq)$. A *multilevel relation with tuple-level labeling* over multilevel relation scheme $(R_i[X_i,K_i],\mathcal{L})$ is a pair (r_i,κ_i), where r_i is a relation over $R_i[X_i,K_i]$ and κ_i is a mapping from tuples over X_i to sets of levels in L, such that $\kappa_i(t) = \{\}$ iff $t \notin r_i$, and $l \in \kappa_i(t)$ if t is labeled at l.

A *multilevel database with tuple-level labeling* over multilevel schema $(\mathcal{B},\mathcal{L})$ is a family $\{(r_i,\kappa_i)\}_{1\le i\le n}$, where (r_i,κ_i) is a multilevel relation with tuple-level labeling over $(R_i[X_i,K_i],\mathcal{L})$. We denote it by the pair (b,κ), where $b = \{r_i\}_{1\le i\le n}$ is a database over $\mathcal{B}$, and $\kappa = \{\kappa_i\}_{1\le i\le n}$ is a family of mappings. Figure 3 shows a multilevel database over the schema of Figure 2 and the lattice of Figure 1. The labels of every tuple are listed to the right of that tuple.

Starship	Mission	Destination	
Enterprise	101	Z	$\top$
Enterprise	102	Rigel	m_1
Enterprise	103	Rigel	m_2
Voyager	102	Rigel	m_1
Voyager	102	Talos	m_2
Discovery	103	Rigel	$\bot$

MissionId	Type	
101	spy	m_1
101	mine	m_2
102	explore	m_1, m_2
103	mine	$\bot$

Figure 3 A Multilevel Database with Tuple-Level Labeling

A multilevel relation (r_i,κ_i) satisfies the *polyinstantiation security property* if:

- for every pair of tuples $t,t' \in r_i$ where $\kappa_i(t) \cap \kappa_i(t') \neq \emptyset$, $t[K_i] = t'[K_i]$ implies $t = t'$.

In other words, tuples labeled at the same level satisfy all the functional dependencies. A multilevel database (b,κ) satisfies the polyinstantiation security property if every multilevel relation in (b,κ) does.

For referential dependency $R_i[Y] \hookrightarrow R_j$, multilevel relations (r_i,κ_i) and (r_j,κ_j) satisfy the *referential security property* if:

- for every tuple $t \in r_i$ and level $l \in \kappa_i(t)$, there is a tuple $t' \in r_j$ and a level $l' \in \kappa_j(t')$ such that $t[Y] = t'[K_j]$ and $l' \preceq^* l$.

In other words, the label of every foreign key tuple dominates the label of the primary key tuple it refers to. A multilevel database (b,κ) satisfies the referential security property if every pair of multilevel relations involved in a referential dependency does. The multilevel database of Figure 3 satisfies polyinstantiation and referential security properties.

3.2 Interpretation Policy for Tuple-Level Labeling

The interpretation policy for tuple-level labeling is straightforward. Because every tuple in a relation represents an elementary fact, the label of the tuple naturally represents the classification of the elementary fact.

Let $(\mathcal{B},\mathcal{L})$ be a multilevel schema, where $\mathcal{B} = (\mathcal{R},\mathcal{C})$, $\mathcal{R} = \{R_i[X_i, K_i]\}_{1\leq i\leq n}$, and $\mathcal{L} = (L, \preceq)$. We can construct a multilevel theory $(\mathcal{L}, \{T^l\}_{l\in L}, \mathcal{A})$ as follows. For every $l \in L$, T^l is the standard first-order interpretation of schema $\mathcal{B}$ where relation scheme R_i is interpreted as predicate R_i^l in the vocabulary of T^l for $1 \leq i \leq n$, and attributes are interpreted as argument positions. Key and referential integrity properties are view constraints and are interpreted as axioms in T^l. For example, the key integrity property over the relation scheme MT of Figure 2 is interpreted as:

$$(\forall x, y, z)(\mathrm{MT}^l(x, y) \wedge \mathrm{MT}^l(x, z) \rightarrow y = z)$$

and the referential integrity property over the schema of Figure 2 is interpreted as:

$$(\forall x, y, z)(\mathrm{SMD}^l(x, y, z) \rightarrow (\exists w)\mathrm{MT}^l(y, w)).$$

Polyinstantiation and referential security properties are labeling constraints and are interpreted as axioms in $\mathcal{A}$. For example, the polyinstantiation security property over the relation scheme MT of Figure 2 and the lattice of Figure 1 is interpreted as:

$$(\forall l \in \mathcal{L})(\forall x, y, z)(\mathrm{MT}^l(x, y) \wedge \mathrm{MT}^l(x, z) \rightarrow y = z).$$

Similarly, the referential security property over the schema of Figure 2 and the lattice of Figure 1 is interpreted as:

$$(\forall l_1 \in \mathcal{L})(\forall x, y, z)(\mathrm{SMD}^{l_1}(x, y, z) \rightarrow (\exists l_2 \in \mathcal{L})(\exists w)(l_2 \preceq l_1 \wedge \mathrm{MT}^{l_2}(y, w))).$$

Let (b, κ) be a multilevel database with tuple-level labeling over $(\mathcal{B},\mathcal{L})$, where $b = \{r_i\}_{1\leq i\leq n}$ and $\kappa = \{\kappa_i\}_{1\leq i\leq n}$. We can construct a multilevel structure of the multilevel theory $(\mathcal{L}, \{T^l\}_{l\in L}, \mathcal{A})$, where $R_i^l(t)$ is true iff $t \in r_i$ and $l \in \kappa_i(t)$.

4 VIEW POLICY

A *view policy* is a specification of the upward information flow—believability of low data at high levels, such that, for every level, view constraints at that level are satisfied with believable data at that level. A view policy should have the following desirable properties:

1. it ensures that believable data do not violate view constraints,
2. it maximizes upward information flow, and
3. it is deterministic.

According to the Bell-LaPadula model, low data can be visible at high. However, since low data could contradict high data in terms of the high view constraints, *visibility* should be distinguished from *believability*.

The filter function (Jajodia, 1990), (Lunt, 1990) and the security logic (Glasgow, 1992) proposed in the literature take one extreme position by equating believability to visibility, thus maximizing believability. However, integrity is compromised if a low tuple contradicts some high tuples with respect to the high view constraints, which leads to an invalid high database. For example, consider the following multilevel relation over the schema of Figure 2 and the lattice of Figure 1:

Starship	Mission	Destination	
Enterprise	$\angle$	Talos	$\top$
Enterprise	102	Rigel	m_1
Enterprise	103	Rigel	m_2

When querying the mission of Enterprise at $\top$, subjects will get back both 102 and 103, which contradicts the view constraint at $\top$ that "starships have unique missions".

Smith and Winslett proposed a belief-based semantics of the multilevel relational model (Smith, 1992), which defines a multilevel relational database as a set of unrelated single-level relational databases, one for every level. They made a clear distinction between visibility and believability, and took the other extreme position by allowing no low tuples to be believable at high, thus minimizing believability. Their semantics provides a foundation based on which other semantics could be compared. However a multilevel relational database that directly employs their semantics would no longer be multilevel—it would be a set of single-level relational databases in which there is no upward information flow cross levels. For example, consider the following multilevel relation over the schema of Figure 2 and the lattice of Figure 1:

Starship	Mission	Destination	
Enterprise	102	Rigel	$\bot$

When querying the mission of Enterprise at $\top$, subjects will get back an empty answer, because no information about Enterprise is considered believable at that level.

The maintenance level of LDV (Haigh, 1991) and the data-based semantics of MLR (Chen, 1995) generalize the belief-based semantics, by allowing users to specify, for every tuple at a level, at which higher levels it is believed. When the view policy for a tuple is missing, the default coincides with the belief-based semantics. Believability is minimized modulo user-supplied exceptions, and there is no automatic upward information flow cross levels. This view policy provides maximal flexibility, at the price that users are burdened with the tedious task of specifying believability for every tuple and every level. For example, to enable subjects at $\top$ to believe whatever information about Enterprise available at $\bot$, the above multilevel relation should be modified to:

Starship	Mission	Destination	
Enterprise	102	Rigel	$\bot, \top$

When querying the mission of Enterprise at $\bot$ or $\top$, subjects will get back 102. However, when querying the mission of Enterprise at m_1 or m_2, subjects still get back an empty answer.

In NTML (Thuraisingham, 1991), Thuraisingham first formalized the distinction between visibility and believability by a proof-theoretic semantics of the multilevel relational model, which consists of a nonmonotonic inference rule stating that low data are believable at high as long as they do not contradict high data. Given two low tuples labeled incomparably, what happens if either tuple does not contradict high data, but their combination does? To determine what is believable at high, the result of Thuraisingham's approach would depend on the (random) order in which the nonmonotonic inference rule is applied to these two tuples, which introduces ambiguity. For example, consider the following multilevel relation over the schema of Figure 2 and the lattice of Figure 1:

Starship	Mission	Destination	
Enterprise	102	Rigel	m_1
Enterprise	103	Talos	m_2

When querying the mission of Enterprise at $\top$, subjects will get back either 102 or 103 but not both.[‡]

In (Qian, 1996), we developed a view policy for multilevel databases with tuple-level labeling, where the view constraints consists of key and referential integrity properties. Informally our view policy states what tuples are believable at a level. First, all tuples labeled at a level should be believable. Second, for tuples labeled at lower levels, as many of them as possible should be believable as long as integrity is preserved, in order to maximize sharing. Third, in case that either but not both of two low tuples could be believable, neither of them should be, because we lack further information to justify the preference of one over the other. In other words, view constraints serve as a filter on how much low data could flow high. For example, consider the multilevel database of Figure 3. When querying the mission of Voyager at $\top$, subjects will get back 102. But when querying the destination of Voyager at $\top$, subjects will get back an empty answer. We can show that our view policy satisfies the three desirable properties identified earlier.

[‡]Notice that such problems occur even with a totally ordered lattice, if we allow arbitrary constraints. For example, a constraint could state that there should be no more than two starships going to Rigel. If we have one tuple (Enterprise, 101, Rigel) at $\top$ together with two tuples (Voyager, 102, Rigel) and (Discovery, 103, Rigel) at $\bot$, then at most one tuple at $\bot$ is believable at $\top$, but it is unclear which one should be.

5 UPDATE POLICY

An *update policy* consists of a set of updates and a specification of the enforcement of labeling constraints on visible data in performing the updates, such that inference channels are eliminated in the enforcement. In particular, if a low update violates labeling constraints when combined with high data, it should not be aborted. Instead it should be extended with necessary compensating high updates in order to enforce integrity, secrecy, and availability. An update policy specifies precisely what the compensating updates should be. It should have the following desirable properties:

1. it does not introduce new inference channels,
2. it does not affect data at lower or incomparable levels,
3. it does not cause denial of service, and
4. it should minimize the effect on data at higher levels.

Lets consider the restricted-value policy of (Sandhu, 1992) and the insert-low policy of (Wiseman, 1990), both of which are designed to enforce no-polyinstantiation.[§] For easy presentation, we adapt these policies to the context of multilevel databases with tuple-level labeling.

Let $(\mathcal{B}, \mathcal{L})$ be a multilevel schema, where $\mathcal{B} = (\mathcal{R}, \mathcal{C})$, $\mathcal{R} = \{R_i[X_i, K_i]\}_{1 \leq i \leq n}$, and $\mathcal{L} = (L, \preceq)$. A multilevel relation (r_i, κ_i) over $(R_i[X_i, K_i], \mathcal{L})$ satisfies the *no-polyinstantiation security property* if:

- for every pair of tuples $t, t' \in r_i$, $t[K_i] = t'[K_i]$ implies $t = t'$.

In other words, tuples labeled at all levels satisfy all the functional dependencies. A multilevel database (b, κ) over $(\mathcal{B}, \mathcal{L})$ satisfies the no-polyinstantiation security property if every multilevel relation in (b, κ) does.

If low subjects insert a tuple which has the same primary key as an existing high tuple, then either the low insertion has to be rejected, causing denial of service to low subjects and leading low subjects to infer the existence of the high tuple, or the high tuple has to be overwritten, causing denial of service to high subjects. Similarly, if high subjects insert a tuple which has the same primary key as an existing low tuple, then either the low tuple has to be deleted, causing denial of service to low subjects and leading low subjects to infer the existence of the high tuple, or the high insertion has to be rejected, causing denial of service to high subjects.

The restricted-value policy and the insert-low policy are very similar. They both remove denial of service to high subjects. The example below illustrates these policies. Consider the following multilevel relation over the schema of Fig. 2 and the lattice of Fig. 1:

[§]No-polyinstantiation can also be enforced by having a more restricted schema, instead of an update policy. For example, the domain of primary key attributes can be partitioned such that primary keys for tuples labeled at different levels are disjoint (Garvey, 1988).

MissionId	Type	
101	explore	$\perp$

Suppose that a $\top$-subject wants to change explore to spy. The restricted-value policy extends the update to:

1. Change explore to $\surd$ at $\perp$.
2. Insert (101, spy) at $\top$.

The extended update ensures no-polyinstantiation at the price of causing denial of service to $\perp$-subjects. The inference channels at update time are replaced by inference channels at query time, because $\perp$-subjects can infer from the restricted-value $\surd$ that mission 101 has a high type. In comparison, the insert-low policy extends the update to:

1. Delete (101, explore) at $\perp$.
2. Insert (101, spy) at $\top$.

The extended update ensures no-polyinstantiation at the price of causing denial of service to $\perp$-subjects. The inference channels at update time remain, because an insertion of mission 101 by $\perp$-subjects will be rejected, leading $\perp$-subjects to infer that mission 101 has a high type.¶

In (Qian, 1994), we showed that no update policies exist for the no-polyinstantiation security property that have all the desirable properties identified earlier, which indicates the inherent difficulty of enforcing unconditional no-polyinstantiation.

In (Qian, 1996), we developed an update policy for multilevel databases with tuple-level labeling, where the labeling constraints consist of polyinstantiation and referential security properties. Informally our update policy states that, if a low tuple to be deleted is referred to by high tuples, the low deletion is extended with a high insertion of a tuple containing the primary key value of the deleted low tuple. For example, deleting the first two MT tuples consecutively (in either order) from the multilevel database of Figure 3 would lead to inserting the MT tuple (101, $\angle$) at $\top$. We can show that our update policy satisfies the four desirable properties identified earlier.

6 CONCLUSION

We have proposed a formal policy framework of MAC policies in multilevel relational databases. We have identified eight components of such policies, and have characterized their desirable properties.

Besides the five components in the traditional interpretation of MAC policies in multilevel databases, one of the most important new components is the interpretation policy. By

¶Both the restricted-value policy and the insert-low policy extend a high update with a low update to eliminate polyinstantiation. In order to avoid covert channels, such extension cannot be done automatically. Rather, it should involve a trusted user performing two separate transactions. In other words, the trusted user should login at high to perform the high update, and login at low to perform the low update.

mapping multilevel relational databases to multilevel theories and structures, the superficial syntactic difference in object labels is abstracted away, and the semantic difference hidden in object labels is made precise. As a consequence, the interpretation policy makes it possible to compare the semantics of multiple MAC policies. As an example, we have developed a natural interpretation policy for multilevel relational databases with tuple-level labeling.

The second new component, the view policy, specifies the upward information flow such that, for every level, view constraints at that level are satisfied with believable data at that level. As an example, we have developed a view policy for multilevel relational databases with tuple-level labeling with three desirable properties, where the view constraints consist of key and referential integrity properties.

The third new component, the update policy, specifies the enforcement of labeling constraints on visible data in performing a set of updates, such that inference channels are eliminated in the enforcement. As an example, we have developed an update policy for multilevel relational databases with tuple-level labeling with four desirable properties, where the labeling constraints consist of polyinstantiation and referential security properties.

The components of a MAC policy could interact in complex ways. For example, the view policy of Trusted ONTOS (Schaefer, 1995) is equivalent to a totally ordered lattice plus NTML (Thuraisingham, 1991). We have also shown in (Qian, 1996) that, for multilevel relational databases with tuple-level labeling, the view policy of (Qian, 1996) applied to the referential integrity property is equivalent to the belief-based semantics (Smith, 1992) applied to the referential security property.

The framework provides a basis for systematically specifying MAC policies and characterizing their potential mismatches. Based on the framework, we have compared and unified the MAC policies and policy components that are proposed in the literature or imposed in existing systems. Our framework could be used to capture and resolve the MAC policy mismatches in the trusted interoperation of heterogeneous multilevel databases. As a first step in this direction, we have investigated the trusted interoperation of multilevel databases whose MAC policies mismatch in the lattice component (Gong, 1994).

ACKNOWLEDGMENT

This work was supported by U.S. Department of Defense Advanced Research Projects Agency and U.S. Air Force Rome Laboratory under contracts F30602-91-C-0092 and F30602-92-C-0140.

REFERENCES

R. K. Burns. Integrity and secrecy: Fundamental conflicts in the database environment. In *Proceedings of the Third RADC Database Security Workshop*, pages 37–40. The MITRE Corporation, 1990.

F. Chen and R. S. Sandhu. The semantics and expressive power of the MLR data model. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, pages 128–42, 1995.

C. Garvey and A. Wu. ASD views. In *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pages 85–95, 1988.

J. Glasgow, G. MacEwen, and P. Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10(3):226–64, August 1992.

L. Gong and X. Qian. The complexity and composability of secure interoperation. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pages 190–200, 1994.

J. T. Haigh, R. C. O'Brien, and D. J. Thomsen. The LDV secure relational DBMS model. In S. Jajodia and C. E. Landwehr, editors, *Database Security, IV: Status and Prospects*, pages 265–79. North-Holland, 1991.

H. H. Hosmer. Integrating security policies. In *Proceedings of the Third RADC Database Security Workshop*, pages 169–73. The MITRE Corporation, 1990.

S. Jajodia and R. Sandhu. Polyinstantiation integrity in multilevel relations. In *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pages 104–15, 1990.

C. E. Landwehr. Formal models for computer security. *ACM Computing Surveys*, 13(3):247–78, September 1981.

T. F. Lunt, D. E. Denning, R. R. Schell, M. Heckman, and W. R. Shockley. The Seaview security model. *IEEE Transactions on Software Engineering*, 16(6):593–607, June 1990.

T. F. Lunt, P. G. Neumann, D. E. Denning, R. R. Schell, M. Heckman, and W. R. Shockley. Secure distributed data views: Security policy and interpretation for DBMS for a class A1 DBMS. Technical Report RADC-TR-89-313, volume 1, Rome Air Development Center, Air Force Systems Command, December 1989.

C. Meadows and S. Jajodia. Integrity versus security in multilevel secure databases. In C. E. Landwehr, editor, *Database Security: Status and Prospects*, pages 89–101. North-Holland, 1988.

J. B. Michael, E. H. Sibley, R. F. Baum, and F. Li. On the axiomatization of security policy: Some tentative observations about logic representation. In B. M. Thuraisingham and C. E. Landwehr, editors, *Database Security, VI: Status and Prospects*, pages 367–86. North-Holland, 1993.

P. Morris and J. McDermid. The structure of permissions: A normative framework for access rights. In C. E. Landwehr and S. Jajodia, editors, *Database Security, V: Status and Prospects*, pages 77–97. North-Holland, 1992.

J.-M. Nicolas and H. Gallaire. Data base: Theory vs. interpretation. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 33–54. Plenum Press, 1978.

G. Pernul. Canonical security modeling for federated databases. In *Proceedings of the IFIP WG 2.6 Conference on Semantics of Interoperable Database Systems*, 1992.

X. Qian. Inference channel-free integrity constraints in multilevel relational databases. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pages 158–67, 1994.

X. Qian and T. F. Lunt. Tuple-level vs. element-level classification. In B. M. Thuraisingham and C. E. Landwehr, editors, *Database Security, VI: Status and Prospects*, pages 301–15. North-Holland, 1993.

X. Qian and T. F. Lunt. A semantic framework of the multilevel secure relational model. To appear in *IEEE Transactions on Knowledge and Data Engineering*, 1996.

R. Sandhu and S. Jajodia. Eliminating polyinstantiation securely. *Computers & Security*, 11(6):547–62, October 1992.

M. Schaefer, P. Martel, T. Kanawati, and V. Lyons. Multilevel data model for the trusted ONTOS prototype. In *Proceedings of the Ninth IFIP WG 11.3 Working Conference on Database Security*, pages 121–41, 1995.

E. H. Sibley, J. B. Michael, and R. L. Wexelblat. Use of an experimental policy workbench: Description and preliminary results. In C. E. Landwehr and S. Jajodia, editors, *Database Security, V: Status and Prospects*, pages 47–76. North-Holland, 1992.

K. Smith and M. Winslett. Entity modeling in the MLS relational model. In *Proceedings of the Eighteenth International Conference on Very Large Data Bases*, pages 199–210, 1992.

G. Steinke and M. Jarke. Support for security modeling in information systems design. In B. M. Thuraisingham and C. E. Landwehr, editors, *Database Security, VI: Status and Prospects*, pages 125–41. North-Holland, 1993.

B. M. Thuraisingham. A nonmonotonic typed multilevel logic for multilevel secure database/knowledge-base management systems. In *Proceedings of the Fourth IEEE Workshop on Computer Security Foundations*, pages 127–38, 1991.

S. R. Wiseman. Control of confidentiality in databases. *Computers & Security*, 9(6):529–37, October 1990.