Proc. 6th RADC Workshop on Multilevel Database Security, Southwest Harbor, Maine, June 1994.

DESIGN AND IMPLEMENTATION OF MULTILEVEL DATABASES

Ravi Sandhu

Department of Information and
Software Systems Engineering
George Mason University
Fairfax, VA 22030-4444
sandhu@isse.gmu.edu

This paper briefly describes ongoing research at GMU on the problem of designing and implementing multilevel databases. In a nutshell the objective of our research is to close the semantic gap between sophisticated requirements of MLS applications and the relatively meager facilities provided by emerging MLS DBMS products. There is a missing links in previous research in the MLS database arena. Previous research has tended to focus either on

- the behavior of a relational MLS DBMS and problems associated with implementing this behavior in different MLS architectures, or
- on stating requirements for an MLS database using semantic data models and related techniques.

There are several notable exceptions to this statement. Sell and Thuraisingham [ST94] have recently proposed a Multilevel Object Modeling Technique (MOMT), patterned on OMT, for designing multilevel database applications using a relational MLS DBMS platform. Lewis and Wiseman [LW93] have also recently described a case study in mapping requirements stated in the SPEAR notation into SWORD and SeaView. The RADC workshop several years ago did a case study of mapping requirements into systems [Smi89, ST89, Hin89, Mai89, Stu89]. The TTCP XTP-1 Workshop on Research Progress in MLS Relational Database held prior to the 1994 RADC workshop also poses a case study.

Our research seeks to reconcile these two streams of activity by addressing the missing-link question of how to achieve the stated requirements on a given data model of MLS relations. It will build upon the prior research cited above. The Sell-Thuraisingham and Lewis-Wiseman efforts were targeted at element-level labeling of MLS relations. Our project will go beyond this work by also considering tuple-level labeling. This is particularly important because emerging MLS DBMS products provide tuple-level labeling rather than the element-level labeling discussed in most of the research literature. This fact widens the semantic gap identified above, and makes the proposed research all the more topical and relevant to the practitioner seeking to build MLS applications on these emerging platforms.

The basic premise of the proposed research is that the theoretical expressive power of fairly simple models can be surprisingly general. The classic example of this is Turing machines, and related automata, which are capable of executing all computable activity. At the same time, simple models are usually not practical to use directly even though they are theoretically capable of solving the problem at hand. It is therefore necessary to develop additional tools to close the semantic gap in a practically useful manner, rather than just declaring theoretical adequacy of a simple model. This approach to closing the semantic gap has been repeatedly employed in computer systems. Our expectation is it will also succeed in the arena of MLS relational databases.

The reason for considering tuple-level labeling is that most of the emerging MLS DBMS products are adopting this approach. This is a natural approach for DBMS vendors, in that the tuple is the basic storage and retrieval unit in typical relational DBMS implementations.

There has been some theoretical discussion of equivalence between tuple-level labeling and element-level labeling. Qian and Lunt [QL93] have published an interesting claim that tuple-level labeling is equivalent to the SeaView model (under a particular definition of equivalence) . We feel this issue needs to be studied more carefully, and in a broader context than the SeaView model. The notion of what is meant by equivalence itself needs a rigorous examination. We now illustrate the subtleties involved here by contrasting two interpretations of tuple-level labeling.

A SIMPLISTIC INTERPRETATION OF TUPLE-LEVEL LABELS

Let us consider a simple mapping from tuple-level labels to element-level labels. Say we have the following tuple

$$(a_1, a_2, ..., a_n, c)$$

where the a_i's are the individual data elements of the tuple, and c is the security label on the tuple. The simplest mapping to element-level labelling is to simply put c as the label of each of the individual elements, as well as let c be the tuple class. This would give us the following tuple (with element-level labeling).

$$(a_1, c, a_2, c, ..., a_n, c_n, c)$$

Each c labels the data element to its left, except for the rightmost one which labels the entire tuple. This simple mapping severely cripples the expressive power of tuple-level labeling. It is impossible to translate the following tuple with element-level labels to an equivalent one with tuple-level labels.

$$(a_1, U, a_2, S, ..., a_n, S, S)$$

This tuple associates an unclassified data element a_1 with a number of secret data elements $a_2 \ldots a_n$. Such an association cannot be expressed with this simplistic interpretation of tuple-level labeling. But this is not the only possible interpretation.

AN ALTERNATE INTERPRETATION OF TUPLE-LEVEL LABELS

Let us consider an alternate interpretation. We caution the reader that this interpretation is being presented only for sake of example. We are not suggesting it as an interpretation to be recommended. Finding useful interpretations of tuple-level labeling is a task for the proposed research.

Let us assume that A_1 (i.e., the first attribute) of a tuple is the apparent key. Now suppose the following tuples are coexisting in a relation (with tuple-level labeling).

$$(a_1, a_2, ..., a_n, U)$$

$$(a_1, a_2', ..., a_n', S)$$

Note that there are two tuples with the same apparent key value (i.e., a_1), so this is a form of polyinstantiation. Now consider the following mapping to element-level labeling.

- Data elements outside of the apparent key inherit the label of the tuple.
- The apparent key is assigned a label equal to the greatest lower bound of the labels of all tuples in which it occurs.

For the pair of tuples shown above we obtain the following two tuples (with element-level labeling) respectively.

$$(a_1, U, a_2, U, ..., U, a_n, U, U)$$

$$(a_1, U, a_2', S, ..., a_n', S, S)$$

Furthermore, consider the following tuple (with element-level labels) which we could not map to an equivalent one with tuple-level labels under the previous interpretation.

$$(a_1, U, a_2, S, ..., a_n, S, S)$$

With the current interpretation we can attempt to translate this into two tuples (with tuple-level labels) as follows.

$$(a_1, null, ..., null, U)$$

$$(a_1, a_2, ..., a_n, S)$$

The first of these essentially fixes the label of a_1 at U. The second gives the S values associated with a_1. Moreover, these two tuples can be interpreted as respectively corresponding to the U and S views of the tuple (with element-level labels) they were derived from.

This alternate interpretation illustrates the important point that it is possible to have a richer semantics for tuple-level labeling than obtained by the simplistic interpretation given earlier. There a number of research questions that need to be addressed here.

Firstly, it is not clear if there is a consistent and useful semantics for tuple-level labeling along the lines sketched out above. The work of Qian and Lunt address this question from one perspective. We feel that a more comprehensive study of this problem is called for, particularly since considerable progress on understanding polyinstantiation has been made in the meantime. Moreover, Qian and Lunt do not consider dynamic aspects of the relations, such as update semantics. In short, much work remains to be done.

Secondly, we must consider the semantics of tuple-level labeling supported in the emerging products. In particular, the update behavior is determined by these DBMS's. It therefore constrains the range of interpretations we can impose on these products.

Thirdly, we must consider how to practically map element-level requirements to tuple-level models. Even if we can establish some kind of theoretical equivalence, we will still need tools and possibly human guidance in achieving an effective mapping.

References

[Hin89] Thomas H. Hinke. Secure database design panel. In *Fourth Annual Computer Security Application Conference*, page 323, Tucson, AZ, December 1989.

[LW93] Sharon Lewis and Simon Wiseman. Database design & MLS DBMSs: An unhappy alliance? In *Ninth Annual Computer Security Application Conference*, pages 232–243, Orlando, FL, December 1993.

[Mai89] Bill Maimone. Oracle corporation homework problem solution. In *Fourth Annual Computer Security Application Conference*, page 324, Tucson, AZ, December 1989.

[QL93] Xiaolei Qian and Teresa Lunt. Tuple-level vs. element-level classification. In B. Thuraisingham and C.E. Landwehr, editors, *Database Security VI: Status and Prospects*, pages 301–315. North-Holland, 1993.

[Smi89] Gary W. Smith. Multilevel secure database design: A practical application. In *Fourth Annual Computer Security Application Conference*, pages 314–321, Tucson, AZ, December 1989.

[ST89] Paul Stachour and Dan Thomsen. A summary of the ldv solution to the homework problem. In *Fourth Annual Computer Security Application Conference*, page 322, Tucson, AZ, December 1989.

[ST94] Peter J. Sell and Bhavani M. Thuraisingham. Applying omt for designing multilevel database applications. In T. Keefe and C.E. Landwehr, editors, *Database Security VII: Status and Prospects*, pages 41–64. North-Holland, 1994.

[Stu89] Edward D. Sturms. Secure database design: An implementation using a secure dbms. In *Fourth Annual Computer Security Application Conference*, page 325, Tucson, AZ, December 1989.