

# Multilevel Databases

Alban Gabillon

Université de Pau et des Pays de l'Adour.

IUT de Mont de Marsan.

LIUPPA/CSySEC

40000 Mont de Marsan, France.

alban.gabillon@univ-pau.fr

## INTRODUCTION

In the context of multilevel security, every piece of information is associated with a *classification* level, and every user is associated with a *clearance* level. The classification and clearance levels are taken from the same set of *security levels*. This set is totally or partially ordered and forms a lattice. The ordering relation is called the *dominance* relation and is denoted by  $\geq$ . An example for a totally ordered set is {Unclassified, Confidential, Secret} with Secret  $>$  Confidential  $>$  Unclassified. An example for a partially ordered set is {low, (Secret, NATO), (Secret, Defence), high} with (Secret, NATO)  $>$  low, (Secret, Defence)  $>$  low, high  $>$  (Secret, NATO), high  $>$  (Secret, Defence), (Secret, NATO) and (Secret, Defence) are incomparable.

In multilevel security, the *security policy* (also called the *confidentiality policy*) says that a user has the permission to *know* a given piece of information only if the clearance level of that user dominates (i.e., is higher than or equal to) the classification level associated with the piece of information. Multilevel security is traditionally opposed to *discretionary security*. In discretionary security, the security rules (permissions or prohibitions) explicitly refer to users' identities.

## BACKGROUND

Most of the existing multilevel security models for databases are based on the following Bell & LaPadula properties (Bell & LaPadula, 1975):

- No Read Up: This rule states that a subject at level X cannot read information at level Y if Y strictly dominates X.
- No Write Down: This rule states that a subject at level X cannot write information at level Y if X strictly dominates Y.

In these rules, *subject* refers to a user or a process. The process level is the *working level* of the user on behalf of which that process executes. The level at which the user decides to work can be his clearance level or any level which is dominated by his clearance level. The No Write Down restriction is necessary to prevent high level processes from illegally disclosing sensitive data. Consider a program which contains a Trojan horse and assume a user with a high clearance decides to work at a high security level and run that program without knowing that it is infected. Without the No Write Down restriction, the Trojan horse would be capable of writing high classified data into a low level information container, making the high level data available to users with low clearance.

The Bell & LaPadula rules do not disallow write up. However, in multilevel databases, write up is very often prohibited because of the integrity problems arising from its blind nature (Thomas & Sandhu, 1993).

The Bell & LaPadula properties are necessary to enforce the confidentiality policy but they are not sufficient. Indeed, it is possible to illegally transmit data by other means than simple read & write operations. In the literature, such unauthorized communication paths are referred to as *covert channels*. Covert channels can be of several types: timing channel, inference channel, signaling channel (National Computer Security Center, 1993).

Multilevel security is for applications requiring a high confidentiality level. It is mainly used by military organizations and it is sometimes called military security.

Many multilevel security models have been proposed for multilevel relational databases (see Denning et al., 1988; Haig et al., 1990; Jajodia & Sandhu, 1991b; Smith & Winslett, 1992; Qian & Lunt, 1996; Sandhu & Chen, 1998; Jukic et al., 1999) and object-oriented databases (see Keefe et al., 1989; Jajodia & Kogan, 1990a; Lunt, 1990; Millen & Lunt, 1992; Cuppens & Gabillon, 1993, 1997, 1999). Our own experience has shown us that when designing a multilevel security model for a database, we need to address the following issues:

- We need to define the granularity of the data. This means we need to identify the data structures which will later be classified. For example, in the context of relational databases, should we assign security levels to tables? to rows? to attributes? to attribute values? Once we have defined the information granularity, we need to specify the semantics of the association between a security level and a granule.
- We need to prevent inference channels. When labelling the data with security levels, we should make sure that no sensitive data can be derived from low level data.
- We need to decompose the multilevel database into a collection of single-level views. Indeed, each user has to be provided with a consistent and complete view of the multilevel database which is compatible with his security level.

- We need to define an operational semantics for the different update operations.

Throughout this paper we shall illustrate these points with a small multilevel relational database reduced to a single relation and we shall consider the following set of security levels {Unclassified (U), Confidential (C), Secret (S)}.

## MULTILEVEL DATABASE DESIGN

### Information Granularity

When designing a multilevel security model for a database, the first problem is to define the information granularity. The finer the information granularity is, the better *the expressive power* of the model is. For example, a model for relational databases where only tables or rows can be classified would be a model with a poor expressive power. Now, a model where attribute values (intersection between a row and a column) can be classified would be a model with a great expressive power.

In fact, defining the information granularity depends on the application needs. In order to illustrate our point, let us consider the `wing` relation (table 1). Primary key of this relation is {Name}.

Table 1: `wing`

Name	Speed	Range	Objective
F16 Falcon	2145 km/h	2000 km	Target A
Mirage 2000	2340 km/h	1480 km	Target B
Rafale	2124 km/h	1824 km	Target A
X-43Z	8000 km/h	9000 km	Target A
Firefox	12000 km/h	13000 km	Target B

This relation represents an imaginary French American wing ready for attack! The military security administrator of this database knows the various sensitivities of the data contained in this table:

- The existence of the wing is unclassified.
- The existence of each plane except the Firefox is unclassified.
- The existence of Firefox is confidential.
- The existence of the Name, Speed and Range attributes in the `wing` table is unclassified.
- The fact that the wing is to launch an attack is confidential, i.e. the existence of the Objective attribute in the `wing` table is confidential.
- The speed and the range of each supersonic plane are unclassified
- The speed and the range of X-43Z are confidential.
- The speed of Firefox is secret and the range is confidential.
- Mirage 2000 and Firefox objectives are secret. Other plane objectives are confidential.

Analyzing the above mentioned facts suggests a model where tables, attributes and attribute values can be classified. The security level assigned to a table protects the existence of the table. The security level assigned to an attribute (column) protects the existence of the attribute in the table. The security level assigned to a primary key value protects the value itself and therefore the existence of the entity which is identified by the value. The security level assigned to a non-key attribute value protects the value itself.

Knowing this, data contained in the wing relation are classified as shown in table 2:

Table 2: Wing (U)

Name (U)	Speed (U)	Range (U)	Objective (C)
F16 Falcon (U)	2145 km/h (U)	2000 km (U)	Target A (C)
Mirage 2000 (U)	2340 km/h (U)	1480 km (U)	Target B (S)
Rafale (U)	2124 km/h (U)	1824 km (U)	Target A (C)
X-43Z (U)	8000 km/h (C)	9000 km (C)	Target A (C)
Firefox (C)	12000 km/h (S)	13000 km (C)	Target B (S)

From this simple example, we learn one important lesson: designing a multilevel security model with a fine granularity is not an insuperable problem provided that the semantics of the association between a granule of information and a security level is precisely defined. This approach was not always followed in the past. For example, *Sea View* (Denning et al., 1988) which is one of the first multilevel security models for relational database assigns security levels to rows as well as to primary key values. Unfortunately, the semantics of such associations is not clearly defined.

## Inference Control

As we said before, the Bell & LaPadula rules are necessary to enforce the confidentiality policy but they are not sufficient. Covert channels can be used to disclose sensitive data. Many types of covert channels cannot be represented in the security model since many of them are due to implementation flaws. However, detecting and eliminating potential inference channels at the model stage is perfectly possible. There is an inference channel when high classified data can be deduced from low classified data. In order to prevent unauthorized inferences, labelling the data with security levels has to be done carefully. For example, the knowledge of “objective is an attribute of the wing table” discloses the knowledge of “the wing table exists”. Consequently, classifying the wing table at a level which strictly dominates the level protecting the objective attribute would create an inference channel. In fact, there should be an *inference control rule* in the security model saying that the security level protecting an attribute has to dominate the security level protecting the table. Another example of potential inference channel is the following: the knowledge of “the speed of Firefox is 6000 km” discloses the existence of Firefox. Consequently, classifying the

primary key value `Firefox` at a level which strictly dominates the level protecting the speed value of `Firefox` would create an inference channel. Therefore, there should be an inference control rule in the model saying that the security level protecting an attribute value of a given row has to dominate the security level protecting the primary key value of that row.

Several of such inference control rules have to be enforced when classifying the data in a relational database. The purpose of this paper is not to discover all these rules. A method to formally derive all these rules as well as one example of application of this method on object-oriented databases is presented in (Cuppens & Gabillon 1998, 1999). Let us mention that this method would allow us to derive one inference control rule from each integrity constraint.

Of course, not all inference channels can be detected and eliminated. In particular, detecting unauthorized inferences from a knowledge which is not represented in the database is impossible.

## **Database Architecture**

The Air Force Summer Study (Air Force Studies Board, 1983) suggested three different architectures for building secure multilevel database management systems. These architectures differ from the way the multilevel data are physically stored. The first architecture is called the kernelized DataBase Management System (DBMS). In this architecture, the multilevel database is partitioned into a collection of single-level segments. The second architecture is called the distributed (or replicated) DBMS. In this architecture, there is a single level database at every security level. Each  $l$ -level database contains the data whose classification level is dominated by  $l$ . The third architecture is based on the integrity lock technology but as mentioned in (Jajodia & Kogan, 1990b) this architecture is vulnerable to Trojan horse. Hence this approach cannot be used for highly-assured multilevel DBMS.

Whether the architecture is kernelized or distributed, we need to decompose the multilevel relation into a collection of single level relations. Such decomposition is not trivial and several algorithms have been proposed in the literature (Denning et al., 1988; Jajodia & Sandhu, 1990a, 1991a). In this paper, we propose some simple guidelines that should be followed for decomposing the multilevel database. These guidelines are suggested in the MultiView model which is presented in (Cuppens & Gabillon 1993, 1997, 1999). This model is based on the replicated architecture. The main advantage of this architecture over the kernelized approach is that there is no need to combine the data from several sources when answering queries. In fact, we claim that the replicated architecture is the best architecture for a fine-grained multilevel database. The principle of the replicated architecture is the following: each granule classified at level  $l$  is stored in the  $l$ -level database and is replicated in every database which has a level dominating

level *l*. Application of this principle to the multilevel `wing` relation (table 2) gives the following three relations (tables 3, 4 and 5):

Table 3: `Wing` (in database at level U)

Name	Speed	Range
F16 Falcon	2145 km/h	2000 km
Mirage 2000	2340 km/h	1480 km
Rafale	2124 km/h	1824 km
X-43Z		

Table 4: `Wing` (in database at level C)

Name	Speed	Range	Objective
F16 Falcon	2145 km/h	2000 km	Target A
Mirage 2000	2340 km/h	1480 km	
Rafale	2124 km/h	1824 km	Target A
X-43Z	8000 km/h	9000 km	Target A
Firefox		13000 km	

Table 5: `Wing` (in database at level S)

Name	Speed	Range	Objective
F16 Falcon	2145 km/h	2000 km	Target A
Mirage 2000	2340 km/h	1480 km	Target B
Rafale	2124 km/h	1824 km	Target A
X-43Z	8000 km/h	9000 km	Target A
Firefox	12000 km/h	13000 km	Target B

The unclassified view (see table 3) of the multilevel `wing` relation contains only unclassified data. This view is inconsistent since there are no values for the speed and the range of X43Z. The confidential view (table 4) of the multilevel `wing` relation contains unclassified and confidential data. This view is inconsistent since there are also some missing values. The secret view (see table 5) of the multilevel `wing` relation contains unclassified, confidential and secret data and is consistent and complete.

After decomposing the original conceptual multilevel relation, the Security Administrator (SA) has basically two solutions to provide unclassified and confidential users with consistent views of the database without disclosing sensitive data:

- The SA thinks that it is acceptable to inform low level users that there are some sensitive values they are not permitted to see. In that case, the SA inserts the special value `RESTRICTED` each time there is a missing value. Using the `RESTRICTED` value was first suggested in (Sandhu & Jajodia, 1992). Semantics of this value is “the value exists but you are not permitted to know it”. Table 6 shows the unclassified view of the `wing` relation after the SA has decided to use the `RESTRICTED` value for the speed and the range of X43Z.

- Now, in some cases, the SA may judge that knowing the existence of the sensitive value is itself sensitive information. In that case, the SA inserts a *cover story* instead of `RESTRICTED`. A cover story is a lie which hides the existence of a sensitive data. Table 7 shows the confidential view of the `wing` relation after the SA has decided to insert a cover story (Target A) each time there was a missing objective value. This means that the SA considers that the existence of a secret objective should not be disclosed. On the contrary, the SA decided to use the `RESTRICTED` value for the speed of Firefox.

Table 6: `wing` (in database at level U)

Name	Speed	Range
F16 Falcon	2145 km/h	2000 km
Mirage 2000	2340 km/h	1480 km
Rafale	2124 km/h	1824 km
X-43Z	<b>RESTRICTED</b>	<b>RESTRICTED</b>

Table 7: `wing` (in database at level C)

Name	Speed	Range	Objective
F16 Falcon	2145 km/h	2000 km	Target A
Mirage 2000	2340 km/h	1480 km	<b>Target A</b>
Rafale	2124 km/h	1824 km	Target A
X-43Z	8000 km/h	9000 km	Target A
Firefox	<b>RESTRICTED</b>	13000 km	<b>Target A</b>

Table 6 is the unclassified view of the multilevel relation represented in figure 2. The users working at the unclassified level express queries on this view. Table 7 is the confidential view on which users working at the confidential level express queries. However, confidential users have also the possibility to query the unclassified view. Table 5 is the secret view. The users working at the secret level express queries on this view, but they may also query the confidential and the unclassified views.

Compared to our approach, other existing decomposition algorithms are complex. This is because these algorithms take *polyinstantiated* multilevel relations as input. A multilevel relation is polyinstantiated if it contains different rows with the same key, each at a different classification level. This occurs when the multilevel relation contains some cover stories.

The advantage of the approach described in this paper is that our starting multilevel relation (table 2) is *not* polyinstantiated. Our starting relation reflects precisely the existing multilevel world. The SA inserts cover stories *after* the decomposition of the starting multilevel relation.

Now, after the insertion of the cover stories, the combination of our three relations (tables 5, 6 and 7) represents a polyinstantiated multilevel world which differs from the original multilevel world. In this polyinstantiated world, the Mirage 2000 and the Firefox have one confidential objective and

one secret objective. The polyinstantiation technique says that in case of conflict, the low classified data shall automatically be interpreted as cover stories. Therefore, if a secret user sees the secret database and the confidential database then he/she shall interpret the confidential objectives of the Mirage 2000 and the Firefox as lies for confidential users.

Note that the polyinstantiation technique has some drawbacks. It relies on a particular interpretation of the data and it does not work well in case of a partial order on the set of security levels. A new technique for managing cover stories is presented in (Cuppens & Gabillon, 2001).

## **Updating the database**

Defining the operational semantics for update operations on a multilevel database is not an easy task. It becomes a real challenge if the multilevel relations are polyinstantiated (see Jajodia & Sandhu, 1990b).

The architecture of our database is replicated. The multilevel world is represented by a set of single level databases. Updating each of these single level databases can be done via standard SQL operations since each single level database behaves as a non protected database. When updating, users interact with the database which corresponds to their working level. The replicated architecture requires that low level updates propagate to the higher levels via a *trusted* replication mechanism. However, this is a general principle which may not be appropriate for some particular situations like the followings:

- How can we interpret the fact that a low level user has updated a low level cover story? Should the update propagate to the higher levels or should the new value be considered as a new cover story?
- How can we interpret the fact that a low level user has inserted a low level row with a primary key value which already exists at higher levels? Should the insertion propagate to higher levels, deleting higher level rows with the same primary key?
- ...

Regarding these issues, there is no best solution. It depends on the integrity policy. With a solution which considers that sensitive data are of high integrity, low level updates should propagate to higher levels as long as they do not conflict with higher classified data. Consequently, the trusted replication mechanisms shall not propagate to higher levels a low level update performed on a `RESTRICTED` value or a cover story. On the opposite, with a solution which emphasizes the freshness of the information, low level updates shall systematically propagate to higher levels, possibly deleting existing higher level data conflicting with the new data. Of course, there are several intermediate solutions. In this paper we suggest one solution which has the advantage of being the simplest: *any low level update statement is reproduced "as such" at the higher levels*. Since each single level database behaves as a non protected database a low level update statement which is

reproduced at a higher level will fail only if it violates one or more integrity constraints.

Let us see the application of our solution through some sample SQL statements:

**INSERT statement.** Consider a user working at the unclassified level who inserts a new aircraft in the unclassified `Wing` relation with the following statement:

```
INSERT INTO Wing VALUES ('F-22 raptor', '2000 km/h', '3200 km');
```

Tables 8, 9 and 10 show the resulting multilevel database.

Table 8: `Wing` (in database at level U)

...	...	...
F-22 raptor	2000 km/h	3200 km

Table 9: `Wing` (in database at level C)

...	...	...	...
F-22 raptor	2000 km/h	3200 km	NULL

Table 10: `Wing` (in database at level S)

...	...	...	...
F-22 raptor	2000 km/h	3200 km	NULL

The replication mechanism has reproduced the `INSERT` statement at the confidential and secret levels. Since the `INSERT` statement did not include any objective value, the confidential and secret objective values are set to `NULL`. If the clearance level of the user who has performed the insertion is at least confidential then that user has now to set his working level to confidential in order to assign an objective to the F-22 raptor. If the clearance level of the user who has performed the insertion is unclassified then only another confidential or secret user may assign an objective to the F-22 raptor.

Propagation of an `INSERT` statement to higher levels will fail only if it violates one or more integrity constraints. For example, if an unclassified user issues a statement inserting a “new” aircraft called Firefox in the unclassified `Wing` relation then the statement will succeed at the unclassified level but fail at the confidential level because of a primary key violation<sup>1</sup>.

**UPDATE statement.** Consider a user working at the confidential level who assigns an objective to the F-22 raptor with the following statement:

```
UPDATE Wing SET Objective='Target A' WHERE name='F-22 Raptor';
```

Table 11 and table 12 show the resulting confidential and secret databases.

---

<sup>1</sup> Of course, whether the propagation of a low level update to higher levels succeeds or fails should not be observable by the low level user who has performed the update.

Table 11: Wing (in database at level C)

...	...	...	...
F-22 raptor	2000 km/h	3200 km	Target A

Table 12: Wing (in database at level S)

...	...	...	...
F-22 raptor	2000 km/h	3200 km	Target A

The replication mechanism has reproduced the `UPDATE` statement at the secret level.

Note that in our example, the replication mechanism would succeed in reproducing a low level update on a `RESTRICTED` value or a cover story since such propagation would not violate any integrity constraint.

**DELETE statement.** Consider a user working at the unclassified level who deletes the F-22 raptor with the following statement:

```
DELETE FROM Wing WHERE name='F-22 Raptor';
```

Tables 5, 6 and 7 show the final multilevel database. The replication mechanism has successfully reproduced the `DELETE` statement at the confidential and secret levels.

Let us mention that the operational semantics for update operations we define in this paper achieves completeness that is, every multilevel database can be constructed by some sequence of update operations (Jajodia & Sandhu, 1991b). However, since every update propagates to the higher levels, single level relations (including `RESTRICTED` values and cover stories) have to be created in the ascending order of the security levels.

## FUTURE TRENDS

In the recent years, research on multilevel security has been eclipsed by research on role-based security (see Sandhu et al., 1996 for an introduction to role-based security). For designing secure commercial database applications, role-based security has proved to be more flexible than multilevel security. Nevertheless, multilevel security is still needed in applications which require a high degree of confidentiality like military applications. One should also mention that research on multilevel security has found new application areas like Digital Rights Management (DRM) architectures for example (see Popescu et al., 2004)

## CONCLUSION

While presenting multilevel security and multilevel databases, we have sketched a multilevel security model for relational databases. Compared to

existing approaches, our model has a better expressive power since tables, attributes and attribute values may independently be classified. Moreover, the semantics of an association between a security level and a granule of information is precisely defined allowing us to easily design conceptual multilevel relations which are not polyinstantiated. The decomposition mechanism we propose is straightforward and is strictly based on the replicated architecture. We use the method developed in (Cuppens & Gabillon 1999) to eliminate inference channels while classifying the data. Finally, we propose a simple operational semantics for the traditional SQL update statements.

## REFERENCES

Bell DE. & LaPadula LJ. (1975). Secure Computer Systems: Unified Exposition and Multics Interpretation. Technical Report ESD-TR-75-306, MTR 2997, MITRE, Bedford, Mass.

Air Force Studies Board (1983). Multilevel Data Management Security. Committee on Multilevel Data Management Security. National Research Council.

Cuppens F. & Gabillon A. (2001). Cover Story Management. Data and Knowledge Engineering, Vol 37/2, pp 177-201. Elsevier.

Cuppens F. & Gabillon A. (1999). Logical Foundations of Multilevel Databases. Data and Knowledge Engineering, vol. 29, pp. 259-291. Elsevier

Cuppens F. & Gabillon A. (1998). Rules for Building Multilevel Object-Oriented Databases. Fifth European Symposium on Research In Computer Security (ESORICS98). Louvain-la-Neuve, Belgium. Lecture Notes in Computer Sciences. Springer Verlag.

Cuppens F. and N., Gabillon A. & Yazdanian K. (1993). MultiView Model for Multilevel Object-Oriented DataBases. Ninth Annual Computer Security Applications Conference. Orlando, USA. IEEE Computer Society.

Cuppens F. & Gabillon A. (1997). A logical Approach To Model A Multilevel Object-Oriented Database. Tenth IFIP 11.3 conference on Database Security. Como, Italy. Database Security, X: Status and Prospects. Pierangela Samarati, Ravi Sandhu (eds.), Chapman & Hall.

Denning DE, Lunt TF., Schell RR., Shockley WR. & Heckman M. (1988). The Sea View Security Model. Proc. of the 1988 IEEE Symposium on Research in Security and Privacy.

Haig JT., O'Brien RC., Stachour PD. & Toups DL. (1990). The LDV approach to database security. In Database Security III, Status and Prospect. North Holland.

Jajodia S. & Kogan B. (1990a). Integrating an Object-Oriented Data Model with Multilevel Security. In IEEE Symposium on Security and Privacy. Oakland 1990.

Jajodia S. & Kogan B. (1990b). Transaction Processing in Multilevel Secure Database Using the Replicated Architecture. Proc. of the 1990 IEEE Symposium on Research in Security and Privacy. Oakland.

Jajodia S. & Sandhu R. (1990a). Polyinstantiation Integrity in Multilevel Relations. IEEE Symposium on Research in Security and Privacy. Oakland. 1990.

Jajodia S., & Sandhu R. (1990b), Update semantics for multilevel relations, in: Proceedings of IEEE Symposium on Research in Security and Privacy, pp. 103-112.

Jajodia S. & Sandhu R. (1991a). A novel decomposition of Multilevel Relations into Single level Relations. IEEE Symposium on Research in Security and Privacy. Oakland. 1991.

Jajodia S. & Sandhu R. (1991b). Toward a multilevel secure relational data model. Proceedings of the 1991 ACM SIGMOD international conference on Management of data.

Jukic N., Vrbsky SV., Parrish AS., Dixon B. & Jukic B. (1999). A belief-consistent multilevel secure relational data model. Information Systems. Vol 24(5). July 1999.

Keefe TF., Tsai WT. & Thuraisingham MB. (1989). SODA: A Secure Object-Oriented Database System. Computer and Security, 8(6).

Lunt TF. (1990). Multilevel Security for Object-Oriented Database Systems. In D.L. Spooner and C. Landwehr editors. Database Security III: Status and Prospects. North-Holland. Result of the IFIP WG 11.3 Workshop on Database Security.

Millen JK. & Lunt TF. (1992). Security for Object-Oriented Database Systems. Proc. of the 1992 IEEE Symposium on Research in Security and Privacy.

National Computer Security Center (1993). A guide to understanding covert channel analysis of trusted systems. November 1993.

Popescu B., Crispo B. & Tanenbaum A.S. (2004). Support for Multi-Level Security Policies in DRM Architectures. New Security Paradigms Workshop. White Point Beach Resort Nova Scotia.

Qian X. & Lunt TF. (1996). A Mac Policy Framework for Multilevel Databases. IEEE Transactions on Knowledge and Data Engineering. Vol 8, Number 1.

Sandhu R. & Chen F. (1998). The MLR Data Model. Transactions on Information and System Security, 1(1):93-132.

Sandhu R., Coyne EJ., Feinstein HL., Youman CE. (1996), "Role-Based Access Control Models", IEEE Computer 29(2): 38-47, IEEE Press.

Sandhu R. & Jajodia S. (1992). Polyinstantiation for cover stories. In European Symposium on Research in Computer Security. Toulouse, France. Springer Verlag.

Thomas RK. & Sandhu R. (1993). Concurrency, Synchronization, and Scheduling to Support High-assurance Write up in Multilevel Object-based Computing. Proceedings of the OOPSLA-93 Conference Workshop on Security for Object-Oriented Systems, Washington DC.

Smith K. & Winslett M. (1992). Entity Modeling in the MLS Relational Model. Proceedings of the 18<sup>th</sup> International Conference on Very Large Data Bases. Vancouver, Canada 1992.

## **TERMS AND DEFINITIONS**

**Multilevel security (MLS):** security where the data are associated with classification levels and the users are associated with clearance levels. Accesses to data are granted by comparing classifications and clearances.

**Classification level:** a security level which represents both the confidentiality degree of the information and its category.

**Clearance level:** a security level that represents both the trust level of the user and his/her need-to-know.

**Information granularity:** refers to the data structure which can be classified.

**Covert channel:** an unintended communications path that can be used to transfer information in a manner that violates the security policy.

**Inference channel:** a particular case of covert channel which exists when high classified data can be deduced from low classified data.

**Cover story:** a lie which is used to hide the existence of a high classified data.