Report No. 90008

Report No. 90008

ROYAL SIGNALS AND RADAR ESTABLISHMENT,
MALVERN

THE SWORD MODEL OF MULTILEVEL SECURE DATABASES

Author: A W Wood

PROCUREMENT EXECUTIVE, MINISTRY OF DEFENCE
RSRE
Malvern, Worcestershire.

June 1990

CONDITIONS OF RELEASE

0077050 BR-114552

************************** DRIC U

************************** DRIC Y

Reports quoted are not necessarily available to members of the public or to commercial organisations.

Royal Signals and Radar Establishment

Report 90008

Title: The SWORD Model of Multilevel Secure Databases

Author: A.W.Wood

Date: June 1990

Abstract

This paper describes SWORD, a new model for a multilevel secure database management system. SWORD provides confidentiality and integrity to *all* users, in a consistent manner without introducing problems of signalling channels and inference. The use of a Trusted Path is shown to allow true multilevel information to be inserted into the database

A-1

1 Introduction

To meet the needs for secure computing in the 1990's, the Information Security Research section at RSRE have embarked on an ambitious project to build a secure computer architecture which aims to achieve the highest possible levels of assurance. This project is called SMITE [Wiseman86, - 88a, Terry89b, Harrold90]. The entire architecture for SMITE, from hardware through microcode to the human interface, is being constructed using rigourous design methodologies. The next stage of the project is to develop a very high assurance secure DBMS based on SMITE principles which will exploit the latter's unique security mechanisms. To this end, the SWORD (SMITE Work On Relational Databases) multilevel secure database model has been conceived out of the original research.

The long-term objective of this development is to design and implement an MLS database system which guarantees confidentiality as well as database integrity to all users, irrespective of their clearance level. The design of SWORD therefore reflects a desire to show all users the true state of the database at all times. In some cases, this involves failing to complete queries posed by users when the query cannot be answered without violating confidentiality.

Such a policy contrasts sharply with the most commonly adopted approach which is known as polyinstantiation [Denning87]. Polyinstantiation achieves confidentiality at the expense of entity integrity. Thus the burden of maintaining this most elementary integrity constraint is carried by the database *applications*, rather than the DBMS itself. SWORD on the other hand provides both confidentiality *and* integrity using a set of controls which can be efficiently implemented with a high degree of assurance. SWORD therefore offers a more suitable platform for applications requiring a multilevel secure database.

As may be expected, data in SWORD is stored in tables made up of a number of *rows* and *columns*. Tables and each of their associated columns are given a name. Within the whole of the SWORD database, tables must have a unique name but column names need be unique only with respect to the table they exist in. These are fairly standard requirements within all database systems. In SWORD however, each table and column is assigned a classification. These *table classifications* and *column classifications* are used to uphold requirements for confidentiality and integrity respectively.

Within each row of a table, there are a number of *fields*. Fields also have a classification associated with them. These *field classifications* are assigned at the time the field is inserted into the SWORD database. Since fields are the basic unit of information, their classifications are the basic mechanism used by SWORD to ensure confidentiality. In SWORD then, a table recording details of aircraft flights and their cargoes may take the following form.

FLIGHTS (CONFIDENTIAL)

FNO	C	PILOT	C	CARGO	S	DESTINATION	TS
123	C	Brown	C	Boots	C	London	C
456	C	Smith	C	Bombs	S	New York	TS
2	C	Jones	C	Engines	C	Paris	S

(Where C denotes CONFIDENTIAL, S denotes SECRET and TS denotes TOP SECRET.The primary key for the FLIGHTS table is FNO, the flight number of the aircraft.)

We will use the above table as an example throughout the remainder of this paper. The following two sections (two and three) respectively highlight the confidentiality and integrity controls of SWORD. In these sections, we will explain how tables such as FLIGHTS are manipulated in a secure manner, and why the classifications of FLIGHTS data are as they are. In section four, we describe how SWORD's confidentiality and integrity controls interact with respect to the simple database operations of insert, delete, select and update. Subsequently, in section five, it will be demonstrated how the key concepts of a general purpose Trusted Path may be used to enhance the accessibility of the SWORD database to a certain class of 'trusted' users. Through the descriptions of both the basic and enhanced controls, the paper shows how the requirement for accuracy and truth in database response can be maintained.

The paper concludes by examining wider, more complex issues which the SWORD research programme is currently pursuing.

2 Confidentiality Controls in SWORD

The primary goal of any multilevel secure database is to maintain the confidentiality of all the information it contains. That is, the security controls within the database must be such that they allow users to gain knowledge of only the information that they are authorised to see - by virtue of their

clearance - and no other. In the SWORD model, this confidentiality is subdivided into two quite separate requirements: no downward flows and no signalling channels.

The simplest example of a downward flow occurs when, in response to a query, a user with clearance w is presented with a field which has a classification v that is not dominated by w (i.e. NOT $w \geq v$). To prevent this flow, SWORD substitutes the value of the sensitive data field with a special "insufficient clearance" value. With respect to downward flows, such 'data replacement' is secure since any user sees only fields with a classification which is dominated by their clearance, or the quite clearly unclassified data "insufficient clearance".

However, masking the output of a query in this way is not enough to achieve complete confidentiality. This is because, in some cases, the actual computation of the result may have depended on the contents of other data fields which the user was not authorised to see. For example, if an CONFIDENTIAL user asked

select PILOT from FLIGHTS where CARGO = Bombs

then in order to return a complete and truthful answer, the database would have to examine a SECRET data field, namely the cargo value for flight 456.

The SWORD solution to this problem is called 'query blocking'. If at any time during the processing of a users query, the database software requires access to a field whose classification dominates the users clearance, then the user is informed that the result of the query may be incomplete. In such cases, the user has the option of continuing or abandoning the query altogether. Although simplistic, such 'query blocking' does nevertheless maintain confidentiality - at no point can the user work out what was in the field which blocked the query. Moreover, the technique allows truthful answers to be obtained at all times even if it is "I cannot fully evaluate the query". In our opinion, this is preferable to the database lying by claiming that the query completed successfully.

'Data replacement' and 'query blocking' are all that is necessary to prevent the direct leakage of information in SWORD. However further controls are required to prevent indirect leakage, that is, via signalling channels. This is because neither 'data replacement' nor 'query blocking' prevent a user from discovering the *existence* of data which was entered by users of higher clearances. High level data is causal to 'query blocking' and, in the case of 'data replacement', each "insufficient clearance" field which appears signals the presence of such data.

To eliminate these signalling problems, SWORD uses table classes in two ways. First, insertion and deletion of rows into a table is permitted only to those users whose clearance equals the table class. Second, only users whose clearance dominates the table class are allowed to perform read and update operations. These controls prevent signalling by ensuring that any user who can detect the existence of data in a table must have a clearance which dominates the clearance of the user who inserted that data. In other words, any signals made through (the existence of) data flow upward, from low to high.

3 Integrity Controls in SWORD

Along with SWORD's controls for confidentiality are several more which ensure the integrity of the database. Integrity in SWORD is maintained through two types of controls - one for the *appropriateness* of the data and its classification and another for their *correctness*.

The appropriateness controls maintain the database in a state which is relevant to the particular present operational requirements. This is achieved by governing the way the SWORD database is modified, that is, by addressing those issues concerning the power of users to do certain actions. For example, to maintain the appropriateness of air force data, we may decide that navy personnel should be prohibited from modifying it. This is a role-based control. There may also though be some cases where it is necessary for army and navy personnel to independently check that a modification is appropriate. An example of this could arise when an aircraft is to be launched off a carrier. This requires separation of duty or n-man rules.

Correctness concerns the validity of the present state of the database. This involves checking not by whom the data is manipulated (as in appropriateness) but what the data actually contains. Such correctness is ensured by adherence to state constraints which describe what data and classifications should be or look like and how they should be inter-related.

All we cover in this paper is the correctness of the classifications and how the confidentiality controls affect the correctness of the data. Appropriateness is essentially application dependent and so its controls reside in a higher level of abstraction than that being described here. We will therefore not

discuss them further, see instead [Terry89a, Harrold90, Wiseman90].

Data typing, entity integrity and referential integrity are some areas in which data correctness criteria appear. For example, it should not be possible to place the character string "XYZ" into the integer FNO column of the FLIGHTS table. Equally, where an entity integrity constraint applies, it should not be possible to insert duplicate flight numbers into FLIGHTS.

In SWORD, the above controls on *data* contents are undoubtedly important. However, it is the correctness of *classifications* which is an imperative requirement for SWORD. Allowing data to be classified too low would render the confidentiality controls ineffective. Classifying data too high on the other hand would lead to a more subtle problem - that of availability.

For SWORD, the integrity control to prevent data being too lowly classified is simply to ensure that users always insert data whose classification dominates their clearance (i.e. 'no write-down'). To maximise availability, SWORD introduces 'column classifications'. Essentially, these are integrity controls on field classes which define an upper bound for the classification of a field in a particular column. This allows SWORD to prevent lowly cleared users interfering with more highly cleared users by adding data with excessively high classifications[1]. For example, if a CONFIDENTIAL user placed a TOP SECRET pilots name into the PILOT column of FLIGHTS, then only those users cleared to TOP SECRET and above will be able to process this column. All other users will be denied access to the TOP SECRET value and thus their queries will be blocked. Making the PILOT column classification be CONFIDENTIAL limits the scope of the lowly cleared user to launch such a serious availability attack.

Column classifications are therefore interesting in their own right, allowing as they do, certain availability guarantees to be made about the behaviour of SWORD for a given query and clearance of user. In conjunction with entity integrity constraints though, they give rise to a particularly important result.

In order to insert into a table which is constrained to have unique keys, the user performing the insertion must be able to examine all the existing keys for possible duplicates. Where this examination is not possible, the insertion must be blocked. Thus, a user can deny all other users the ability to insert by adding key data with a high classification. Consequently, to maintain entity integrity *and* availability, the SWORD database designer will probably ensure that users add key data with a classification equal to their clearance. This can be simply enforced by making all key column classifications equal to the table classification.

4 Data Operations

In sections two and three, we have seen that SWORD is a multilevel secure database model which addresses the core security issues of confidentiality, availability and integrity. SWORD's field classifications are used to prevent downward flows of information whilst table classifications eliminate signalling channels. Column classifications guarantee that the availability of the database is not compromised by the overclassification of data.

In this section, we will describe how the basic database operations of insert (a row), delete (several rows), update (several fields) and select (several rows) are implemented in SWORD. Each of these operations requires a different combination of SWORD's security controls. However, it will be shown that each combination gives reliable and truthful answers to users of sufficient clearance.

4.1 Insert

To insert a new row in SWORD,a user specifies a sequence of (field, field class) pairs and a target table. Thus, for example, the query

insert ("789" C, "Wilson" S, "Medicines" U, Beirut C) into FLIGHTS

may be used to record the assignment of a mercy flight to Beirut. Note that here users are assumed to specify all values and classifications to fit the target table. Some top-level applications may however allow users to specify only a few of the required fields, allowing a default mechanism to fill in the rest. At the level of SWORD being described in this paper though the effect of this is not visible and all fields (even nulls) are present.

When a new row is specified for insertion, SWORD ensures four things:

[1]Conceptually , column classifications could also be used to prevent interference from highly cleared users. This is discussed in section five.

- there is no insecure flow through the signalling channel
- there are no downward flows
- field class integrity is maintained
- integrity constraints (such as entity integrity and data typing) are upheld

The control for signalling is the simple one given in section two - only users whose clearance equals the classification of the target table may perform the insert operation. The controls for the next two requirements are also straightforward. For each (field, classification) pair, SWORD checks that the classification part dominates the user's clearance but is dominated by the column classification for the field. If any of these checks fails, the operation is rejected and the user informed of the reason. Likewise, the insertion is rejected if any integrity constraints are violated.

By the nature of all of these controls, the information contained in any rejection message that may occur can not in anyway violate confidentiality. This is because the user must be aware of their own clearance and knowledge of column classifications in SWORD is classified at the level of the table. Thus any of the warnings

> SECRET user cannot insert into CONFIDENTIAL table FLIGHTS
> CONFIDENTIAL user cannot insert UNCLASSIFIED data
> SECRET value in PILOT column of FLIGHTS not permitted
> String type "789" is the wrong type for FNO column of FLIGHTS

convey no extra (classified) knowledge to the user.

4.2 Delete

In SWORD, the signalling problem through delete operations is handled identically to that for insertion - users may only attempt to delete rows in a table whose table class equals their clearance. Again, the only possible signal is from low to high.

Given that a user has the correct clearance to perform a delete operation, SWORD then proceeds to execute the query, blocking this execution process as necessary. Such blocking would occur only when the user specifies a 'where clause' which is based on a column that contains highly classified fields. For example, the following query, from a CONFIDENTIAL user,

> delete all from FLIGHTS where CARGO = Bullets

would be blocked by the need to examine the SECRET cargo field containing "Bombs". In this case the user will be told that no rows were deleted but that one row was skipped because it had a highly classified cargo.

Note though that this technique does not prevent the removal of rows containing data classified higher than the user's clearance. Thus the query,

> delete all from FLIGHTS where FNO = 456

would be executed successfully on behalf of a CONFIDENTIAL user, despite the fact that the SECRET cargo of flight 456 would be deleted. This is an issue of integrity, not one of confidentiality, and is therefore dealt with by the separate integrity controls supported by SWORD.

4.3 Select

To read the data of a table in SWORD, a user specifies a 'select' query. This is a simple type of query which returns all the data from the target table that satisfies the user's selection criteria (from the 'where' clause). Since no data modification is involved, select queries in SWORD cannot drive a signalling channel nor, by virtue of the table classification control, can they be used to receive signals. Therefore, any user with a clearance which dominates the table classification can request a select operation.

Selection can however give rise to the same confidentiality problems as those of delete through the use of accessing columns of data. For this reason, we apply identical controls: a user will be returned only those rows which were not blocked out. This control does not depend on the list of columns which are to appear in the output. For instance, the query

> select FNO, DESTINATION from FLIGHTS where PILOT <> King

requires only access to PILOT data in building its response, not the FNO or DESTINATION columns.

and will therefore be completed for a CONFIDENTIAL user.

The confidentiality of the fields which appear as the output of a select query is ensured by the technique of 'data replacement' - "insufficient clearance" replacing all fields whose classification strictly dominates the clearance of the user. For a CONFIDENTIAL user then, the result of the above query would be:

FNO	C	DESTINATION	TS
123	C	London	C
456	C	Ø	TS
2	C	Ø	S

(where Ø denotes "insufficient clearance").

4.4 Update

The update operation is the most complex operation of the four in this section. In SWORD, both fields and classifications may be altered by such an operation. The controls to ensure the security of an update request must thus embrace all the aspects of confidentiality and integrity.

To prevent signalling, SWORD applies controls on the ability of users to change the classifications of fields since this is the only method of driving a signalling channel through updates. The action of this control is dependent on the clearance of the user performing the update and the classification of the target table. Only those users whose clearance equals the table classification can change a field's classification. Since no user cleared lower than the table classification can access the table and thus see the effect of this change, there is no channel.

As in select and delete, updates must address the 'no downward flow ' aspect of confidentiality. There are two sides to this control. In the first instance, all fields which need to be examined as the query is being processed must have a classification which is dominated by user's clearance. Following on from this, any new classifications which are assigned during the update must dominate both the old classifications and the user's clearance. This last is simply the obvious control to prevent unauthorised downgrading. In SWORD, the update is blocked if any one of these conditions is not met.

Column classifications are used to limit availability attacks through updates. As a field's classification is altered, it must be checked against the relevant column classification. Once more, the update is blocked if 'field class integrity' is violated. Equally, an update operation is blocked if it would violate any other integrity constraints like those on unique primary keys

5 SWORD and the SMITE Trusted Path

As described in this paper, SWORD allows only those users cleared to exactly the classification of a certain table to insert and delete rows from that table. Also, users may only change the classifications of fields during an update if their clearance equals the table classification. In all cases, sections two and four have shown this to be a secure policy with respect to signalling channels, since all signals flow from low to high. However, in some applications, it may be unacceptable to restrict the ability to insert and delete data in this manner. By relying on a general purpose Trusted Path mechanism, such as that of the SMITE secure capability computer [Wiseman88b, Wood88], these restrictions can be removed, making the SWORD model even more powerful.

A Trusted Path is essentially a user interface which acts faithfully on behalf of human users. Normally, the Trusted Path of a secure system is extremely simple and is used only for such elementary functions as logging in. The SMITE Trusted Path, on the other hand, is a truly universal interface, with all the functionality of traditional, non-secure WIMP systems. This allows it to be used for *all* interactions with the SMITE secure system, not just those considered to be security critical.

In SMITE each program callable from the Trusted Path must be either 'trusted' or 'untrusted'. 'Trusted' software is an extension of the Trusted Path, that is, it acts faithfully, doing only the tasks it is actioned to do. Such software is deemed not to contain Trojan Horse code which exploits signalling channels. During evaluation, 'trust' is given to a program by some validation or verification process which checks for undesirable features. 'Untrusted' software, on the other hand, consists of those programs which have not been evaluated. We must therefore assume that they do contain Trojan Horse software which sends or receives signals. Most user developed programs or off-the-shelf packages fall into this latter category.

Given this division of trust between software in SMITE, it becomes easy to define when a query addressed to the SWORD database is not being used as part of a signalling channel. Only when the

entire execution path of the query, from the top level query specification to the physical disc access to the processing of the answer, consists entirely of 'trusted' software can we be sure that no channels are being exploited. This allows us to relax some of the restrictions on which users may insert and delete rows. For instance, a trusted user with a high clearance can be permitted to delete rows in a table with a low table classification.

Further relaxations are possible if the Trusted Path allows the user to manipulate fields of various classifications, as is the case for SMITE. A Trusted Path of this nature can be used to construct multilevel rows for insertion into the SWORD database. The signalling channels which exist through the data fields and their classifications are not a problem because the software is trusted not to exploit them. For example, suppose a SECRET user retrieves a number of UNCLASSIFIED fields from the database using the Trusted Path. Subsequently, the user may decide to insert one of these fields back into the database, again using the Trusted Path. Although inserted by a SECRET user, the new field can remain UNCLASSIFIED, there being no explicit downward flow since the Trusted Path has ensured that the field was not modified with highly classified information. Equally, because the user is trusted, there can be no signal made through the choice of which UNCLASSIFIED field was inserted.

6 Further Remarks

The SWORD model is based on a single, sound premise: that the users of a multilevel secure database system should be able to rely on the database to give them full and correct answers, without violating any data integrity or security policy requirements. In this paper, we have shown that it is possible to achieve such a state with no significant loss of functionality: SWORD's table and column classifications can guarantee that the database always remains accessible to some users and constraints on the data may be maintained at all times. We have further shown, through the use of a Trusted Path mechanism, that this functionality within SWORD can be enhanced. Simply by assigning 'trust' to certain user processes, we can add more flexibility to the SWORD model.

In order to continue building on the foundation described in the paper, much diverse work is being done on SWORD by the Information Security Research Section. The ultimate goal of this research is the implementation of a full SWORD database system on SMITE which meets the highest possible levels of assurance.

Parts of the SWORD model have already been incorporated into a front-end for commercially available, non-secure database management packages. The aim of this work is two-fold. First, of course, by making available a relatively simple method of building a relational database, we can better appreciate the difficulties that will be involved in constructing the full SWORD implementation. Second, and more importantly, we can demonstrate the applicability of SWORD-based controls to existing databases. This front-end may form the basis of a low assurance security upgrade path for some organisations.

As the first steps towards a full, high assurance implementation of SWORD, a formal specification of its security controls is being written using the Terry-Wiseman security model [Terry89a]. In tandem with this, research is also being undertaken to specifying the functionality properties of a high-assurance system, from the physical disc handling aspects to the top level user interface. This latter piece of work will cover such problem areas as transaction management, concurrency controls (locking) and rollback and recovery. Many of these lower level details require especially careful consideration due to their potential for being exploited as signalling channels. Although the rigourous design will not be completed until the early part of 1991, some Ada implementation code will have been produced before this date. This code will be used to confirm that some key ideas of SWORD are practical.

At a more abstract level, much modelling work has been done on quantifying inference in multilevel database systems in general [Cordingley89, Sowerbutts90]. The basis of this work is that an MLS database can be viewed as a set, or collection, of classified facts. Work on the facts model has been useful in determining how security affects the data design process. It has also shown how SWORD may better use its concept of column classifications - by making them 'content-dependent' we can achieve a significant reduction in the amount of queries that need be blocked. For example, at present, the query

select PILOT from FLIGHTS where CARGO = Engines

would be blocked for a user whose clearance does not dominate SECRET because a full answer can only be given by examining a SECRET cargo field. If however, the column classification of the PILOT column were SECRET for Boots and Bombs but only CONFIDENTIAL for Engines then we can place an integrity constraint on any user who loads a plane with engines, namely that the cargo must be marked at or below CONFIDENTIAL. The above query will now be able to proceed when requested by a CONFIDENTIAL user. By looking up the column classification for Engines, SWORD can determine that the above query will be correctly and completely answered by examining only cargo fields

classified at or below CONFIDENTIAL.

7 Conclusions

SWORD is a powerful yet simple model of a multilevel secure relational database. SWORD is unique in that it provides confidentiality without compromising data integrity. In particular, signalling channels are not admitted and yet entity integrity is strictly enforced. As such, all users of SWORD are presented with a consistent view of the database.

By employing a Trusted Path mechanism, such as that in the SMITE secure computer, SWORD can be used in a true multilevel fashion. For example, rows made up of data which is genuinely of different sensitivity levels may be both inserted and retrieved.

8 References

[Cordingley89] S. Cordingley and B.J.Sowerbutts, "The Fact Database - A Formal Model of the Inference Problem for Multilevel Secure Databases", *Plessey Report 72/89/R420/U*, November 1989.

[Denning87] D.E.Denning et al., "A Multilevel Relational Data Model", *Proceedings of 1987 IEEE Symposium on Security and Privacy*, Oakland, California, pp. 220-234, April 27-29, 1987.

[Harrold89] C.L.Harrold, "An Introduction to the SMITE Approach to Secure Computing", *Computers and Security*, Vol. 8, No. 6, pp. 495-505, October 1989.

[Harrold90] C.L.Harrold, "The Terry-Wiseman Security Policy Model and Examples of its Use", *RSRE Report 90001*, March 1990.

[Sowerbutts90] B.J.Sowerbutts and S.P.Cordingley, "A Model of Inference Control in a Multilevel Secure Database Management System", *Plessey Report 72 90 R173 U*, April 1990.

[Terry89a] Phil Terry and Simon Wiseman, "A 'New' Security Model", *Proceedings of 1989 IEEE Computer Society Symposium on Security and Privacy*, Oakland, California, pp. 215-228, May 1-3, 1989.

[Terry89b] P.F.Terry, "The SMITE Approach to Security", *RSRE Report 89014*, August 1989.

[Wiseman86] Simon Wiseman, "A Capability Approach to Multi-Level Security", *Proceedings of IFIP/Sec'86 International Conference on Computer Security*, Monte Carlo, Monaco, December 1986.

[Wiseman88a] S.R.Wiseman, "Protection and Security Mechanisms in the SMITE Capability Computer", *RSRE Memorandum 4117*, January 1988.

[Wiseman88b] Simon Wiseman, Phil Terry, Andrew Wood and Clare Harrold, "The Trusted Path between SMITE and the User", *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, Oakland, California, pp. 147-155, April 18-21, 1988.

[Wiseman90] Simon Wiseman, "The Control of Integrity in Databases", to appear in *Proceedings of Fourth IFIP WG 11.3 Workshop on Database Security*, Halifax, England, September 18-21, 1990.

[Wood88] A.W.Wood, "A Z Specification of the MaCHO Interface Editor", *RSRE Memorandum 4247*, November 1988.

REPORT DOCUMENTATION PAGE

DRIC Reference Number (if known) .....

Overall security classification of sheet .....Unclassified.....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the field concerned must be marked to indicate the classification eg (R), (C) or (S).

Originators Reference/Report No.	Month	Year
REPORT 90008	JUNE	1990

Originators Name and Location
RSRE, St Andrews Road
Malvern, Worcs WR14 3PS

Monitoring Agency Name and Location

Title
THE SWORD MODEL OF MULTILEVEL SECURE DATABASES

Report Security Classification	Title Classification (U, R, C or S)
UNCLASSIFIED	U

Foreign Language Title (in the case of translations)

Conference Details

Agency Reference	Contract Number and Period

Project Number	Other References

Authors	Pagination and Ref
WOOD, A W	8

Abstract

This paper describes SWORD, a new model for a multilevel secure database management system. SWORD provides confidentiality and integrity to *all* users, in a consistent manner, without introducing problems of signalling channels and inference. The use of a Trusted Path is shown to allow true multilevel information to be inserted into the database.

Abstract Classification (U,R,C or S)
U

Descriptors

Distribution Statement (Enter any limitations on the distribution of the document)
UNLIMITED