# Secure Concurrency Control Algorithm for MultiLevel Secure Databases

**Sonakshi Shanwal[1], Suresh Kumar[2]**

*[1,2]Department of CSE, Faculty of Engineering and Technology, MRIU, Faridabad*
*[1]sonakshi.shanwal@gmail.com, [2]suresh.fet@mriu.edu.in*

**ABSTRACT**
In a Multi Level Database (MLS) data as well as user, both are classified in order to provide security to data. Data and users are classified at different levels in the database and the user with a particular security level is allowed to access the data at that level or below that level only. So the concurrency control requirements of MLS databases are different from the concurrency control requirements of traditional databases. In this paper we have analysed the problems that may occur when conventional methods of concurrency control are used in MLS databases. We have proposed secure and starvation-free concurrency control algorithm.

**KEYWORDS**
Concurrency Control, Databases, Multilevel Secure Databases.

## 1. INTRODUCTION

Like traditional databases, Multi Level Database (MLS) databases are also used by multiple users at the same time. The MLS databases are shared by concurrent transactions with different classification levels. Certain security aspects are imposed to traditional databases to convert them into Multi Level Database. Some multilevel secure database models use access-control protocols based on the Bell-LaPadula model [7]. It has two properties: simple security property and star property. The first property is for read-access and the second property is for providing write-access. With these new constraints the traditional concurrency control techniques are not suitable for MLS databases because some additional inconsistencies occur due to them. Since the objective of multilevel secure databases was to classify data as well as user to enhance the security, it may be violated if the traditional techniques of concurrency control are used [1, 9].

Two main issues that may arise due to the basic approaches of concurrency control are the inference problem and signalling channels. These are the major threats to design of multilevel secure databases and are discussed in detail in the following section of the paper. To overcome these issues, many researchers have proposed secure algorithms for concurrency control in multilevel databases [8]. In this paper a starvation-free secure concurrency control algorithm is proposed.

## 2. SECURITY ISSUES

In the terminology of MLS databases an object may be a data file, a record or a field within a record, and, a subject is an active entity that can request for read/ write access to the objects. In the MLS databases both objects and subjects are classified. The security levels of the objects are called classification levels and that of the subjects are termed as clearance levels. The combination of classification levels and the clearance levels is called a label. The MLS databases are based on Bell-LaPadula security model which have the following properties [7]:

Simple Security Property: It says that a subject is allowed to have read-access to an object only if the clearance level of the subject is identical to or higher than the classification level of the object.

Star Property: According to this property, a subject is allowed to have a write-access on an object only if the clearance level of the subject is identical to the classification level of the object.

These two properties make sure that information do not flow from higher security level object to lower security level subjects. However, these protocols prevent the direct flow of information, but there is a possibility that there may be an indirect flow of information from the subjects at higher level to the subjects at lower level subject through covert channels. If a communication channel is not designed or intended to transfer information from one level to another level but it does then it is called a covert channel. The covert channels are classified into two categories: storage covert channels and timing covert channels. A covert channel is a storage covert channel if it involves the direct storage location of other entity. A timing covert channel is one in which a transaction classified at higher level signals information to another transaction classified at lower level by modulating its own use of systems resources in such a manner that the real response time observed by the lower level transaction is affected by this manipulation[15]. For example, when a user classified at a lower security level want to insert some

data and that data already present in the database at a higher level of security. When this insert operation is rejected by the system, then the lower level user will get to know that same data already exist in the system at a higher security level. This indirect flow of information from higher security level to lower security level is possible through different ways. For instance, the concurrent execution of transactions may lead to contention of data objects. If the results from a lower security level transaction are delayed, when a higher security level transaction is executing, then the user at lower security level can determine the presence of higher security level transactions, and may be able to infer some meaningful information by interpreting the length of the delay. In concurrency control approaches covert channels are generally established when a resource or a data object is shared between the subjects with different classification levels [8].

The inference channel in a database is a method by which the users classified at lower security levels can infer data classified at higher levels. Hence the objective is to detect and remove inference channels. When traditional approaches for concurrency control like locking techniques and time stamping are applied to multilevel databases, channels are established between the transaction at low level and the transaction at high level as discussed in [1,8].

## 3. LITERATURE REVIEW

In a multilevel secure database, security is imposed at different levels. In the literature the architectures of these databases are categorised into two broad categories [8, 15]:Woods-Hole Architecture and Trusted Subject Architecture. In Woods-Hole architecture security is provided by the underlying operating system whereas in Trusted subject architecture both the database and the operating system are responsible for security.

N. Dobrinkova [7] describes LaPadula Bell-model. The model deals with the control of information flow and is a linear non-discretionary model. This model of protection consists of the following components: A set of subjects, a set of objects, an access control matrix, and several ordered security levels. This model was first published in 1987 and it was a proposal for enforcing access control in government and military applications. This model has a set of four access-rights: Read-Only, Append, Execute, and Read-Write. These accesses refer to the operations with the subjects and to enforce data security and integrity by imposing Reading-down and Writing-up restrictions.

D. E. Denning et. al. [3] proposed a model called the SeaView model. This was developed by SRI International and Gemini Computers in 1985. In this model, several policies were developed in order to enforce mandatory access control (MAC), discretionary access controls (DAC) and relational integrity constraints, for multilevel databases. SeaView model is a formal security model that combines, through its policies, software and hardware, in order to provide data security to multilevel databases.

S. Jajodia et. al. [12] gave an orange locking protocol for concurrency control in MLS databases. In an orange locking protocol when a transaction $T_L$ classified at low clearance level tries to write a data object x at same level classification; while a transaction $T_H$ classified at high clearance level has already acquired a read lock on that data object. At this point, the read-lock granted to the transaction $T_H$ is converted to an orange lock. By this the contents of the data-object x is invalidated by other transaction's write operation. At the commit phase of transaction $T_H$, it checks its entire write operation of transaction $T_L$, to see if there is any read-lock converted into an orange lock. If any orange lock is present, $T_H$ may be aborted, or rolled-back and then executed again starting from the firstly invalidated data object after $T_L$ finishes. Therefore, the creation of a covert channel is prevented using orange locks.

H. T. Kim et. al. [5] proposed a secure concurrency control protocol. They presented a concept of invisible area and t-locks. The invisible area of a high transaction $T_i$ is a time interval for which transaction $T_i$ is blocked by any other lower transaction $T_j$. The purpose of defining the invisible area is to hide the operations of lower level transactions from the currently blocked high transactions and prevent transaction $T_j$ from reading new versions of data objects created by lower level transaction $T_j$ running within this area; otherwise $T_j$ may suffer from a retrieval anomaly when it resumes its execution.

N. Kaur et. Al. [8] proposed a multi-version concurrency control algorithm. They have changed the condition for a transaction to be included in the conflict set of any other transaction. In the algorithm read-set of transaction R-set$T_j$ is divided into two parts R-set$_{doneTj}$ and R-set$_{remainingTj}$. By this modification they were able to improve degree of concurrency but it may lead to retrieval anomaly.

In this case, older versions of the conflicting variables are used by higher level transaction, however new versions of these variables are created by the lower level transactions. As a result recent data is not reflected to high level transaction. Another issue was that no factor was included in the algorithm to claim it to be starvation-free. A higher transaction may be blocked infinite number of times and hence the higher transaction is starved. This algorithm can be explained with the help of following example. Suppose there are two transactions $T_H$ and $T_L$. The transaction $T_H$ represents transaction of a user classified at higher security level and $T_L$ belong to a user that is classified at lower classification level.

$T_H$: r[x0] r[y0], w[y1], c;

$T_L$: r[x0], w[x1], c;

When $T_H$ was executing and transaction $T_L$ with lower classification level enters into the system and since it is of lower classification level, transaction $T_L$ is made to wait. Due to this timing channel may be established as the transaction at lower classification level can observe the time interval when it is forced to wait. In order to avoid this problem of signalling channel $T_H$ is blocked and $T_L$ is executed. Transaction $T_H$ resumes its execution when transaction $T_L$ commits, but changes made by $T_L$ are not reflected to $T_H$ . In other words $T_L$ is executed in the invisible area of transaction $T_H$.

## 4. PROPOSED WORK: STARVATION-FREE SECURE CONCURRENCY CONTROL ALGORITHM

If we add the concept of orange locks to the algorithm proposed by H. T. Kim [5], it will remove the retrieval anomaly. The above algorithm is modified and the concept of invisible area is removed as the changes made by a lower transaction must be visible to a higher transaction in any case to remove retrieval anomaly. In order to make the algorithm starvation-free we attach a counter with every transaction and when this counter crosses the maximum limit the transaction will be executed and the lower transactions will have to wait. The algorithm will proceed in the following way:

1: The scheduler receives counter P (with initial value, say n) read-set R-Set $T_i$ and0 write-set W-Set $T_i$ when transaction $T_i$ is submitted.

2: When there is no transaction in execution, the scheduler executes transaction $T_i$. When transaction $T_i$ commits go to 6.

3: When there is a transaction $T_j$ that is currently in execution and another transaction $T_i$ arrives, the scheduler can take three decisions according to the security level of the incoming transaction, $T_i$:

i) If $L(T_i) > L(T_j)$. Go to **4**.

ii) If $L(T_i) = L(T_j)$. Go to **5**.

iii) If $L(T_i) < L(T_j)$.

If $R\text{-set}_{Tj} \cap W\text{-set}_{Ti} \neq \varphi$

Then transaction $T_j$ is blocked by the scheduler and transaction $T_i$ will start executing. When transaction $T_i$ commits, transaction $T_j$ resumes its execution, but it immediately check whether transaction $T_j$ has acquired any orange locks. If found, transaction $T_j$ is rolled back to release these locks and executed again.

If $R\text{-set}_{Tj} \cap W\text{-set}_{Ti} = \varphi$

Both transactions will be executed simultaneously without disturbing each other.

4: Since a low level transaction $T_j$ has been running, the scheduler makes the high level transaction $T_i$ wait until $T_j$ terminates. When $T_j$ commits go to 6.

5: Because both the transactions are at the same security level, the scheduler execute them concurrently.
If R-setTj $\cap$ W-setTi $\neq \phi$,

Then at the time of conflicting data item, transaction $T_j$ is blocked by the scheduler.

Let transaction $T_i$ will execute only for the duration of creating new versions of conflicting data objects. When transaction $T_j$ resumes its execution it is checked if any orange locks has been acquired by the transaction, if found, transaction $T_j$ is rolled-back. When both the transactions $T_i$ and $T_j$ commits, go to 6.

6: When transaction $T_i$ was the only transaction to be executed at 2, the scheduler waits for other transactions to be submitted. Otherwise, the scheduler will select a transaction (say $T_k$) with lowest security level among the blocked transactions, along with new transactions with the same security level as that of transaction $T_k$.

## 5. ILLUSTRATIVE EXAMPLE

Let there are two transactions $T_H$ and $T_L$ where $T_H$ is the transaction of user classified at higher level i.e. one whose classification level is higher and $T_L$ is the transaction of user with lower classification level.

| $T_H$ (High Transaction) | $T_L$ (Low Transaction) |
|---|---|
| r[x0]<br>r[y0],<br>w[y1],<br>c; | <br>r[x0],<br>w[x1],<br>c; |

Here, $r[x_i]$ represents read operation on $i^{th}$ version of data object 'x' and similarly $w[x_i]$ is the write operation on $i^{th}$ version of data object 'x' and 'c' represents commit operation.

According to the previous algorithms When transaction $T_L$ enters the system the scheduler blocks the high transaction and the low transaction is made to execute in the invisible area of high transaction. After the low transaction commits the high transaction resumes its execution, but it is unaware of the changes made by the low transaction as changes by low transaction are invisible to high transaction. Due to this, retrieval anomaly exists i.e. a most recent value of any item is not retrieved by the blocked high transaction.

With the proposed algorithm this anomaly can be removed. When $T_L$ commits and $T_H$ resumes execution, it is immediately checked if transaction has acquired any orange lock. If so the transaction is rolled-back in order to release this orange lock otherwise executed.

Example 2: Let there be another set of transactions $T_1$, $T_2$, and $T_3$ in increasing order of levels i.e. $T_1$ has highest classification and $T_3$ has the lowest level.

When transaction $T_2$ is entered transaction $T_1$ is currently executing. Since it has lower clearance level, so transaction $T_1$ is blocked and transaction $T_2$ starts its execution. Now transaction $T_3$ enters the system which is having the lowest classification level among all the active transactions. At this moment transaction $T_2$ will also be blocked and transaction $T_3$ will starts its execution. When transaction $T_2$ resumes its execution, the scheduler checks if transaction $T_2$ has acquired any orange lock on data object 'y', as 'y' is modified by transaction $T_3$, after it is read by transaction $T_2$. So it is changed into orange lock and the transaction $T_2$ is rolled-back and it is executed again with the recent version of 'y'. When it completes its execution transaction $T_1$ is checked for any orange lock, as data object 'x' has been modified by transaction $T_2$, after it is read by transaction $T_1$. So transaction $T_1$ is also rolled-back and executed again with the recent version of 'x'.

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| r[x0] | | |
| | r[x0], | r[y0], |
| | w[x1], | w[y1], |
| | r[y0] | c; |
| r[y0], | r[xo], | |
| w[y1], | w[x1], | |
| c; | r[y1], | |
| | r[z0], | |
| | w[z1], | |
| | c | |

## 6. CONCLUSIONS

We have analysed that the traditional approaches for concurrency control cannot be applied to multilevel secure databases and also studied the proposed algorithms for same in multilevel secure databases. We proposed a new algorithm for starvation-free concurrency control in MLS databases. In our algorithm we included the concept of orange locks. In future performance of this algorithm can be evaluated by practically imposing it on the multilevel secure database for concurrency control because there exists a trade-off as when we make the algorithm starvation-free, covert channels may be established and when we remove the possibility of covert channel, it will no longer remain starvation-free.

## 7. REFERENCES


[1] A.K. Khan, "Meeting Security Requirements on Transaction Processing in MLS Databases", Minnesota State University, Mankato, USA, March, 2013.

[2] B. Panda, W. Perrizo, R. Haraty, "Secure Transaction Management and Query Processing in Multilevel Secure Database Systems", Proc. of ACM Symposium on Applied Computing, pp: 363-368, 1994.

[3] D. E. Denning, T. F.Lunt. "The SeaView Security Model", IEEE Transactions On Software Engineering, vol. 16, issue 6, pp: 593-607, 1988.

[4] E. Bertino and R. Sandhu, "Database security-Concepts, Approaches and Challenges", IEEE Transactions on Dependable Secure Computing", vol. 2, issue 1, pp: 2-19, 2005.

[5] H. T. Kim and M. H. Kim, "Starvation-Free Secure Multiverison Concurrency Control", Information Processing Letters, Vol. 65, pp. 247-253, 1998.

[6] L. V. Mancini and I. Ray, "Secure Concurrency Control in MLS Databases with Two Versions of Data", Computer Security, Vol. 1146, pp: 304-323, 1996.

[7] Nina Dobrinkova, "Information Security- Bell-LaPadula Model. Institute of Information and Communication Technologies" Sofia, 2010.

[8] N. Kaur, R. Singh, M. Misra& A. K. Sarje, "Concurrency Control for Multilevel Secure Databases", International Journal of Network Security, Vol.9, No.1, PP.70-81, July 2009.

[9] N. Kaur, R. Singh, M. Misra and A. K. Sarje, "Performance Evaluation of Secure Concurrency Control Algorithm for MLS Databases", Proceedings of International Conference on Information Technology, IEEE, 2005.

[10] R. Elmasri, S. Navathe,"Fundamentals of Database Systems", ISBN 0-201-54263-3, Addison-Wesley, 2000.

[11] S. Kang, S. Moon. "Read Down Conflict-Preserving Serializibility as a Correctness Criterion for Multilevel

Secure Optimistic Concurrency Control", CRD, Journal of System Architecture, pp. 889-902, 2000.

[12] S. Jajodia, L. V. Mancini and I. Ray, "Secure Locking Protocol For Multilevel Database Mangement Systems", Proceedings of IFIP Conf. Database Security, pp: 177-194, 1997.

[13] S. Jajodia, C. McCallum,"Using Two Phase Commit for Crash Recovery for Federated Multilevel Secure Database Systems", Dependable computing Fault Tolerant Systems, vol. 8, pp. 365-381, New York, Springer, Verlag, 1993.

[14] V. Atluri, S. Jajodia, E. Bertino, "Alternative correctness criteria for concurrent execution of transactions in MLS databases", IEEE Trans. Knowledge and Data Engineering, pp: 839-854, 1996.

[15] W. Rjaibi, "An Introuction to Multilevel Secure Relational Database Management Systems", Proceedings of Center for Advanced Studies on Colaaborative Research, pp. 232-241, 2004.