Improving Timeliness in Real-Time Secure Database Systems

Sang H. Son*, Rasikan David*, and Bhavani Thuraisingham†

*Department of Computer Science
University of Virginia
Charlottesville, VA 22903

†Mitre Corporation
Bedford, MA 01730

ABSTRACT

Database systems for real-time applications must satisfy timing constraints associated with transactions, while maintaining data consistency. In addition to real-time requirements, security is usually required in many applications. Multilevel security requirements introduce a new dimension to transaction processing in real-time database systems. In this paper, we argue that because of the complexities involved, trade-offs need to be made between security and timeliness. We briefly present the secure two-phase locking protocol and discuss an adaptive method to support trading off security for timeliness, depending on the current state of the system. The performance of the adaptive secure two-phase locking protocol shows improved timeliness. We also discuss future research direction to improve timeliness of secure database systems.

1. Introduction

Database security is concerned with the ability of a database management system to enforce a security policy governing the disclosure, modification or destruction of information. Many secure database systems use an access control mechanism based on the Bell-LaPadula model [Bell 76]. This model is stated in terms of subjects and objects. An object can be a data file, record or a field within a record. A subject is an active process that requests access to objects. Every object is assigned a classification and every subject a clearance. Classifications and clearances are collectively referred to as security classes (or levels) and they are partially ordered. The Bell-LaPadula model imposes the following restrictions on all data accesses:

a) *Simple Security Property*: A subject is allowed read access to an object only if the former's clearance is identical to or higher (in the partial order) than the latter's classification.

b) *The *-Property*: A subject is allowed write access to an object only if the former's clearance is identical to or lower than the latter's classification.

Database systems that support the Bell-LaPadula properties are called multilevel secure database systems (MLS/DBMS). The Bell-LaPadula model prevents direct flow of information from a higher access class to a lower access class, but the conditions are not sufficient to ensure that security is not violated indirectly through what are known as covert channels [Lamp 73]. A covert channel allows indirect transfer of information from a subject at a higher access class to a subject at a lower access class. An important class of covert channels that are usually associated with concurrency control mechanisms are timing channels. A timing channel arises when a resource or object in the database is shared between subjects with different access classes. The two subjects can cooperate with each other to transfer information.

A real-time database management system is a transaction processing system where some transactions have explicit timing constraints [Son 95b]. Typically a timing constraint is expressed in the form of a deadline, a certain time in the future by which a transaction needs to be completed. As advanced database systems are being used in applications which need to support timeliness while managing sensitive information, one cannot avoid the need for integrating real-time data processing techniques into MLS/DBMSs. For certain applications in which absolute security is required for safety-critical operations, any trade-offs of security for timeliness cannot be allowed. The approach presented in this paper is not intended to cover such applications.

Concurrency control is used in databases to manage the concurrent execution of operations by different subjects on the same data object such that consistency is maintained [Bern 87]. In multilevel secure databases, there is the additional problem of maintaining consistency without introducing covert channels. For a more detailed description of and a possible solution to the problem of concurrency control in secure databases, the reader is referred to [Dav 93], [Thur 93]. In this paper, we discuss the additional issues that arise when transactions in a secure database have timing constraints associated with them. We first review related work in secure concurrency control, and discuss the

This work was supported in part by ONR.

problems associated with time-constrained secure concurrency control. An adaptive method by which security requirements can be partially compromised for improved timeliness is then presented.

2. Background

Covert channel analysis and removal is one of the important issues in multilevel secure concurrency control. The notion of *non-interference* has been proposed [Gogu 82] as a simple and intuitively satisfying definition of what it means for a system to be secure. The property of *non-interference* states that the output as seen by a subject must be unaffected by the inputs of another subject at a higher access class. This means that a subject at a lower access class should not be able to distinguish between the outputs from the system in response to an input sequence including actions from a higher level subject and an input sequence in which all inputs at a higher access class have been removed. An extensive analysis of the possible covert channels and the necessary and sufficient conditions for a secure, interference-free scheduler are given in [Keef 90].

Locking will fail in a secure database because the security properties prevent actions in a transaction T_1 at a higher access class from delaying actions in a transaction T_2 at a lower access class (e.g. when T_2 requests a conflicting lock on a data item on which T_1 holds a lock). Timestamp ordering fails for similar reasons, with timestamps taking the role of locks, since a transaction at a higher access class cannot cause the aborting of another transaction at a lower access class. Locking and timestamping techniques can be adapted for MLS/DBMSs.

Optimistic concurrency control for a secure database can be made to work by ensuring that whenever a conflict is detected between a transaction T_h at a higher access class in its validation phase and a transaction T_l at a lower access class, the transaction at the higher access class is aborted, while the transaction at the lower access class is not affected. A major problem with using optimistic concurrency control is the possible *starvation* of higher-level transactions. For example, consider a long-running transaction T_h that must read several lower-level data items before the validation stage. In this case, there is a high probability of conflict and as a result, T_h may have to be rolled back and restarted an indefinite number of times.

A secure version of the MVTO scheduler is presented in [Keef 90b]. The difference between Basic MVTO and Secure MVTO is that Secure MVTO will sometimes assign a new transaction a timestamp that is earlier than the current timestamp. This effectively moves the transaction into the past with respect to active transactions. This method has the drawback that transactions at a higher access class are forced to read arbitrarily old values from the database due to the timestamp assignment. This problem can be especially serious if most of the lower level transactions are long running transactions. Alternative approach is to make higher access class transaction wait until all transactions that are lower and have arrived earlier finish their execution [Jajo 92].

3. Supporting Security and Timeliness

There are several papers that have explored approaches to extend conventional databases for time-critical applications [Abbo 92], [Hari 90], [Lee96], [Sha 91], [Son 92]. The problem arises when these approaches are applied to secure databases, because covert channels can be introduced by priority based scheduling. All existing real-time systems schedule transactions based on some priority scheme. The priority usually reflects how close the transaction is to missing its deadline. Priority-based scheduling of real-time transactions, however, interacts with the property of non-interference which has to be satisfied for security. For example, consider the following sequence of requests:

T_1 (SECRET)	:	*R(X)*			
T_2 (UNCLASSIFIED)	:		*W(X)*		
T_3(UNCLASSIFIED)	:			*W(X)*	
T_4(UNCLASSIFIED)	:				*R(X)*

Assume that T_1, T_2 and T_3 have priorities *5, 7* and *10* respectively and the priority assignment scheme is such that if *priority*(T_2) > *priority*(T_1), then T_2 has greater criticalness and has to be scheduled ahead of T_1. In the above example, T_2 and T_3 are initially blocked by T_1 when they arrive. When T_1 completes execution, T_3 is scheduled ahead of T_2, since it has a greater priority than T_2 and the transaction execution order would be T_1 T_3 T_2 T_4. However, if the transaction T_1 is removed, the execution order would be T_2 T_3 T_4 because T_2 would have been scheduled as soon as it had arrived. The presence of the SECRET transaction T_1 thus changes the value read by the UNCLASSIFIED transaction T_4, which is a violation of *value security.* For the same reason *delay security* is also violated, because the presence of T_1 delays T_2 with respect to T_3.

From this example, it is clear that priority-based transaction scheduling is not feasible for a fully secure database system. It is because in a secure environment, a transaction at a higher level:

- cannot cause the aborting of a transaction at a lower access class. If it is allowed to do so, it is possible that it can control the number of times a lower level transaction is aborted, thereby opening a covert channel.
- cannot conflict with a transaction at a lower access class. If such a conflict does occur, the higher level transaction has to be blocked or aborted, not the low level

transaction.

- cannot be granted greater priority of execution over a transaction at a lower access class.

Therefore, for minimizing deadline miss percentage, we take the approach that partial security violations under certain conditions are permissible.

4. Secure Two-Phase Locking

Basic two-phase locking does not work for secure databases because a transaction at a lower access class (say T_l) cannot be blocked due to a conflicting lock held by a transaction at a higher access class (T_h). If T_l were somehow allowed to continue with its execution in spite of the conflict, then non-interference would be satisfied. We have developed a secure two-phase locking protocol to solve this problem [Son 94]. The basic principle behind the secure two-phase locking protocol is to try to simulate execution of Basic 2PL without blocking of lower access class transactions by higher access class transactions. Three different types of locks are used for this purpose. Their semantics are explained below:

1) *Real Lock (of the form $pl_i[x]$):* A real lock is set for an action $p_i[x]$ if no other conflicting action has a real lock or a virtual lock on x. The semantics of this lock are identical to that of the lock in basic two phase locking.

2) *Virtual Lock (of the form $vpl_i[x]$)*: A virtual lock $vpl_i[x]$ is set for an action $p_i[x]$ if a transaction at a higher access class holds a conflicting lock on x ($p_i[x]$ has to be a write to satisfy the Bell-LaPadula properties). The virtual lock is non-blocking. Once a virtual lock $vpl_i[x]$ is set, $p_i[x]$ is added to *queue[x]* and the next action in T_i is ready for scheduling. When $p_i[x]$ gets to the front of the lock queue, its virtual lock is upgraded to a real lock and $p_i[x]$ is submitted to the scheduler. A virtual lock holding action $vpl_i[x]$ can be superseded in the lock queue by a conflicting action $q_j[x]$ if T_j is in *before(T_i)*.

3) *Dependent Virtual Lock (of the form dvpli[x])*: A dependent virtual lock is set for an action $p_i[x]$ (where p is a write) if a previous write $w_i[y]$ in the same transaction holds a virtual lock. An action $p_i[x]$ which holds a dependent virtual lock with respect to another action $w_i[y]$ is not allowed to set a real lock or a virtual lock unless $w_i[y]$'s virtual lock is upgraded to a real lock. The dependent lock is non-blocking and can be superseded by a conflicting action $q_j[x]$ if T_j is before T_i in the serialization order.

A more detailed description of the secure two-phase locking protocol is given in [Son 94].

5. An Adaptive Security Policy

Results from performance analysis of Secure 2PL exhibit a better response time characteristic than Secure OCC. Its operating region (the portion of the curve before the saturation point) is much larger than that of Secure OCC. Further, staleness is not an issue in Secure 2PL as with Secure MVTO. However, this alone does not suffice when timing constraints are present on transactions. In Secure 2PL, transaction scheduling order is determined purely by the order in which transactions acquire locks. No conscious effort is made to schedule transactions according to their priority, or according to how close a transaction is to meeting its deadline. In a real-time database system this is unacceptable. Therefore, security properties may need to be compromised to some extent to ensure a certain degree of deadline cognizance.

A covert timing channel is opened between two collaborating transactions - one at a higher access class and the other at a lower access class - if the higher access class transaction can influence the delay seen by a lower access class transaction. The *bandwidth* of a covert channel is a measure of how easy it is for the higher access class transaction to control the delay seen by the lower access class transaction. If there is a great degree of randomness in the system, i.e., an indeterminate number of transactions could be affecting the delay that the higher access class transactions wants a lower access class transaction to experience, then the bandwidth is low. On the other hand, if the higher access class transaction knows that the lower access class transaction to which it wants to transmit information is the only other transaction in the system, then the bandwidth is infinite. Therefore, when security has to be sacrificed, a policy that keeps the bandwidth of the resulting covert channel to a minimum is desirable. To ensure this, the security policy has to be adaptive, i.e., determining whether security is to be violated or not when a conflict arises should depend on the current state of the system and not on a static, predecided property.

Our adaptive policy to resolve conflicts between lock holding and lock requesting transactions is based on past execution history. Whenever a transaction T_1 requests a lock on a data item x on which another transaction T_2 holds a conflicting lock, there are two possible options:

- T_1 could be blocked until T_2 releases the lock.
- T_2 could be aborted and the lock granted to T_1.

If T_1 were at a higher security level than T_2, the latter option would be a violation of security. However, if T_1 has greater priority than T_2, then the latter option would be the option taken by a real-time concurrency control approach. In our approach, we strike a balance between these two conflicting options by looking up past history. A measure of the degree to which security has been violated in the past is calculated. A similar measure of the degree to which the real-time constraints have not been satisfied can be obtained from the number of deadlines missed in the past. These two measures are compared and depending on which value is greater, either the security properties are satisfied or the higher priority transaction is given the right to execute.

The two factors that are used to resolve a conflict are:

- Security Factor (SF):
  (number of conflicts for which security is maintained/ total number of conflicts) * difference in security level between the two conflicting transactions.
- Deadline Miss Factor (DMF):
  number of transactions that missed their deadline/total number of transactions committed

Two factors are involved in the calculation of SF. The first factor is the degree to which security has been satisfied in the past, measured by the number of conflicts for which security has been maintained. Secondly, we also assume that the greater the difference in security levels between the transactions involved in the conflict, the more important it is to maintain security. DMF is determined only by the number of deadline misses in the past. Note that for a comparison with DMF, (1 - SF) has to be used, since (1 - SF) is a measure of the degree to which security has been violated. Now, a simple comparison $(1 - SF) > DMF$ is not enough, since different systems need to maintain different levels of security. Therefore, we define two weighting factors, α and β for (1 - SF) and DMF respectively. If $\alpha * (1 - SF) > \beta * DMF$, then for the conflict under consideration, the security properties are more important and therefore the conflict is decided in favor of the transaction at a lower access class. If the opposite is true, then the transaction with higher priority is given precedence. Note that at low conflict rates, it is possible to satisfy both the security and the real-time requirements simultaneously. As a result the comparison is not made until the DMF reaches a certain threshold value DMISS_THRESH. The parameters DMISS_THRESH, α and β can be tuned for the desired level of security. A very high value of DMISS_THRESH or a very high value of α compared to β would result in SF being maintained at 1.0, i.e., for all conflicts the security properties are satisfied. A very high value of β compared to α would result in an SF value of 0.0, i.e., the behavior would be identical to that of 2PL-HP [Abbo 92]. For a desired value of SF between 0 and 1, the values of α, β and DMISS_THRESH would have to be tuned based on the arrival rate of transactions.

The adaptive protocol can be described by the rules specifying how to resolve conflict.

If a conflict between a lock holding transaction T_1 and a lock requesting transaction T_2 arises, the conflict is settled using the following rules:

```
• If DMF < DMISS_THRESH
  then
      follow the steps taken by the Secure 2PL protocol
• Else If $\alpha * (1 - SF) > \beta * DMF$
      follow the steps taken by the Secure 2PL protocol
• Else
      break the conflict in favor of the transaction with
      the higher priority
```

The performance results of the adaptive secure 2PL protocol for a spectrum of security factor values are reported in [Son 95].

6. Conclusions

In this paper, we have presented an approach to scheduling transactions to improve timeliness in a secure real-time database. The performance results substantiate our claim that an adaptive security policy that sacrifices the security properties to some extent can improve the deadline miss performance.

The work described in this paper is more a direction for future research than a concrete solution to the problem of secure real-time concurrency control. There are a number of issues that need to be looked into. First of all, a proper characterization of the bandwidth of a covert channel that can arise given a particular value of SF needs to be derived. Applications might express a desired level of security in terms of a maximum admissible bandwidth of a potential covert channel. Unless there is a way of determining to what extent a security policy satisfies the security properties, one cannot determine whether the policy is suitable for the application or not. We have investigated how to model the bandwidth of a covert channel, based on information theory which is concerned with the possibility of noise degrading the fidelity of the signal being transmitted [Dav 95]. We are currently pursuing mechanisms to control the bandwidth (call capacity) of covert channels.

Secondly, in this paper we have considered a simple trade-off between deadline miss percentage and security. A trade-off could also have been made between alternative factors depending on the application. Thirdly, we have restricted ourselves to a soft deadline system with no overload management policy. It would be interesting to see how a policy to screen out transactions that are about to miss their deadline would affect performance.

Finally, in this paper, we have restricted ourselves to the problem of real-time secure concurrency control in a database system. Some of the other issues that need to be considered in designing a comprehensive real-time multilevel secure database system (MLS/RTDBMS) are discussed in [Son 93]. Various types of MLS/RTDBMSs need to be identified and architectures and algorithms developed for each type of system. Trade-offs need to be made between security, timeliness and consistency on a case-by-case basis.

References

[Abbo 92] Robert K. Abbott and Hector Garcia-Molina. "Scheduling Real-Time Transactions: A Performance Evaluation", ACM Transactions on Database Systems, Vol. 17, No. 3, pp 513-560, September '92.

[Bell 76] D. E. Bell and L. J. LaPadula. "Secure Computer Systems: Unified Exposition and Multics Interpretation", The Mitre Corp., March 1976.

[Bern 87] P. A. Bernstein, N. Goodman & V. Hadzilacos. "Concurrency Control and Recovery in Database Systems", Reading, MA: Addison Wesley, 1987.

[Dav 93] Rasikan David & Sang H. Son. "A Secure Two Phase Locking Protocol", Proceedings of the 12th Symposium on Reliable Distributed Systems, Princeton, NJ, October 1993.

[Dav 95] Rasikan David, Sang H. Son, and Ravi Mukkamala. "Supporting Multilevel Security Requirements in Real-Time Databases", IEEE Symposium on Security and Privacy, Oakland, CA, May 1995, pp 199-210.

[Gogu 82] J. A. Goguen and J. Meseguer. "Security Policy and Security Models", Proceedings of the IEEE Symposium on Security and Privacy, pp 11-20, 1982.

[Gree 91] Ira Greenberg. "Distributed Database Security", Technical Report A002, SRI International, April 1991.

[Hari 90] Haritsa, J. R., M. J. Carey and M. Livny. "Dynamic Real-Time Optimistic Concurrency Control", Proceedings of the IEEE Real-time Systems Symposium, Orlando, FL, Dec '90.

[Jajo 92] Sushil Jajodia & Vijayalaksmi Atluri. "Alternative Correctness Criteria for Concurrent Execution of Transactions in Multilevel Secure Databases", Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, May 1992.

[Keef 90] T. F. Keefe, W. T. Tsai & J. Srivastava. "Multilevel Secure Database Concurrency Control", In Proceedings of the Sixth International Conference on Data Engineering, pp 337-344, Los Angeles, CA, February 1990.

[Keef 90b] T. F. Keefe & W. T. Tsai. "Multiversion Concurrency Control for Multilevel Secure Database Systems", Proceedings of the 11th IEEE Symposium on Security and Privacy, Oakland, California, pp 369-383, April 1990.

[Lamp 73] Butler W. Lampson. "A Note on the Confinement Problem", Communications of the ACM, Vol. 16, No. 10, pp 613-615, October 1973.

[Lee 96] Juhnyoung Lee and Sang H. Son. "Concurrency Control Algorithms for Real-Time Database Systems" in "Performance of Concurrency Control Mechanisms in Centralized Database Systems", V. J. Kumar (editor), Prentice Hall 1996, pp 429-457.

[Sha 91] Sha L., R. Rajkumar, S. H. Son, C. Chang. "A Real-Time Locking Protocol", IEEE Trans. on Computers, Vol. 40, No. 7, July 1991.

[Son 92] Sang H. Son, J. Lee & Yi Lin. "Hybrid Protocols Using Dynamic Adjustment of Serialization Order for Real-Time Concurrency Control", Real-Time Systems Journal, Vol. 4, No. 3, pp 269-276, September 1992.

[Son 93] Sang H. Son and Bhavani Thuraisingham. "Towards a Multilevel Secure Database Management System for Real-Time Applications", IEEE Workshop on Real-Time Applications, New York, New York, May 1993.

[Son 94] Sang H. Son and Rasikan David. "Design and Analysis of a Secure Two Phase Locking Protocol", International Computer Software and Applications Conference (COMPSAC'94), Taipei, Taiwan, pp 374-379, November 1994.

[Son 95] Sang H. Son, Rasikan David, and Bhavani Thuraisingham. "An Adaptive Policy for Improved Timeliness in Secure Database Systems", IFIP WG 11.3 Conference of Database Security, Rensselaerville, New York, Aug. 1995, pp 223-233.

[Son 95b] Sang H. Son. "Real-Time Database Systems: Present and Future," International Workshop on Real-Time Computing Systems and Applications, Tokyo, Japan, Oct. 1995.

[Thur 93] Bhavani Thuraisingham and Hai-Ping Ko. "Concurrency Control in Trusted Database Management Systems: A Survey", SIGMOD RECORD, Vol. 22, No. 4, December 1993.