Proceedings of the IBIMA 2005 Conference. Lisbon, Portugal. 2005.

Towards a Temporal Multilevel Secure Relational Database Model

Ramzi A. Haraty and Natalie Bekai
Lebanese American University
Beirut, Lebanon
Email: rharaty@lau.edu.lb

Abstract

Conventional temporal databases provide little support for ensuring secrecy of sensitive data. On the other hand, multilevel secure relational data models assure that each user gains access to only those data for which he/she has proper clearance, but there is no support for recording and querying time varying data. In this paper, we aim to provide a new relational database model that supports both recording and ensuring the secrecy of time varying data.

Keywords: Multilevel secure databases, polyinstantiation, and temporal databases.

1. Introduction

It is important before starting to talk about temporal multilevel secure relational data model that we understand the classical multilevel secure model and temporal relational database model. In this paper, we review the basic concepts of both - the multilevel secure database model and the temporal database model, their definition, and the essential constraints that must be satisfied.

Numerous works have been done on temporal and multilevel secure database models [2, 3, 5, 6, 7, 8]. However, none of them combined both issues under one model. In this paper, we aim to provide a new relational database model that supports both recording and ensuring the secrecy of time varying data.

The remainder of the paper is organized as follows: section 2 discusses the multilevel secure database model. Section 3 overviews the temporal database model. Section 4 presents our combined model and section 5 concludes the paper.

2. Multilevel Secure Databases

Multilevel secure databases are databases that contain large amounts of very highly sensitive and confidential data (e.g., military, governmental, etc.); that is why access to the data stored in these databases needs to be authorized. Although, there is no clear agreement on the definition of a multilevel secure database model, we try to present in this section the basic concepts of a multilevel secure relational model. Our aim is to use its fundamental aspects in building up the model of temporal multilevel secure databases.

Basic Concepts

One of the main concepts in multilevel secure databases is the assignment of access privileges to users of the database so as to be able to manage and protect confidential and sensitive data. Each user is given access privileges to access the data he/she is authorized to access. Confidential data is protected either by making it inaccessible to unauthorized or by providing a cover story. To provide a cover story, the same real-world entity is depicted by more than one record. Each of these records is assigned a different classification level. Users with different access clearances see different versions of the data in the database. These records have the same primary key at all the classification levels but with different values for the non-key attributes at each classification level [1]. This technique is used to protect information stored at a higher security levels by providing some lower security levels. Data hidden from low clearance users will be seen by a user of a higher clearance if this user has the clearance to see this data.

Access privileges can be assigned to relations, to individual tuples in a relation, to individual columns, or to individual data elements of a relation. In this paper we assume access classes are assigned to individual data elements of a relation.

Multilevel Relations

The definition of multilevel relation is divided into two parts:

Part 1 [RELATION SCHEME] A state-invariant multilevel relation scheme is of the form:
$R(A_1,C_1,A_2,C_2,\ldots,A_n,C_n,TC)$
Where A_i is a data attribute over domain D_i, C_i is a classification attribute for A_i, and TC is the tuple-class attribute. The domain of C_i is specified by a set $\{L_i, \ldots, H_i\}$, which enumerates the allowed values for access classes, ranging from the greatest lower bound (glb) L_i to the least upper bound (lub) H_i. The domain of TC is the set $\{lub\{Li; i=1,\ldots,n\},\ldots, lub\{Hi: i= 1,\ldots,n\}\}$.

Part 2 [RELATION INSTANCES] A collection of state-dependent relation instances are of the form
$R_c(A_1,C_1,A_2,C_2,\ldots,A_n,C_n,TC)$
with one instance present for each class c in the given lattice. Each instance is a set of distinct tuples of the form $(a_1,c_1,a_2,c_2,\ldots,a_n,c_n,tc)$ where each $a_i \in D_i$, or a_i = null, $c \geq c_i$ and tc = lub$\{c_i : i = 1 \ldots n\}$. Moreover, if a_i is not null then ci $\in [L_i, H_i]$, which means that a classification attribute can not be null. Each instance in a multilevel relation is supposed to present the version of reality appropriate for each access class.

To further clarify the multilevel relation definition, let us take for example the military officers relation presented in figure 1.1 and figure 1.2. The relation scheme for the relation military officers is presented below:
Military Officers (ID, Name, Empl_Date, Rank)
The military classifies its database users into two clearances categories: U, and S. The users at the U level will be able to see only the data that they have been given the clearance, while the S level users will be able to see the data stored at the U level along with the data stored at the S level. In figure 1 we see the U-user version of military officers' relation, and in figure 2 we see the S-user version of military officers' relation. The information about the military officer whose ID no. 101 can not be seen by a U user since it has been assigned a higher access clearance, therefore this information can only be seen by a user who has a security level S or higher. In this example, we see also that the information about the military officer no. 100 at the U level is different from that at the S level. As we can see, the true identity and rank of the officer 100 has been masked by a cover story. The U level users are given a cover identity and rank for the officer 100.

ID	Name	Empl_Date	Rank	TC
100 U	Johnson U	1953 U	Major General U	U

Military_Officers$_u$

Figure 1. The U-User version of the military officers table.

ID	Name	Empl_Date	Rank	TC
100 U	Johnson U	1953 U	Major General U	U
100 U	William S	1953 U	Inspector General S	S
101 S	Miles S	1934 S	Inspector General S	S

Military_Officers$_S$

Figure 2. The S-User version of the military officers table.

3. TEMPORAL DATABASES

Temporal databases are used in environments where special support for the storage and querying of historical and future data is a requirement. A temporal database is a database that contains not only current data but also historical data, and even possibly future data. Conventional databases by contrast contain only current data.

BASIC CONCEPTS

A standard relation is two-dimensional with attributes and tuples as dimensions. A temporal relation contains two additional, orthogonal time dimensions, namely valid time and transaction time. Valid time denotes the time period during which a fact is true with respect to the real world. Transaction time records when facts are stored in the temporal relation. Valid and transaction time have precise, crisp definitions. If changes to the past are important, then valid time support is required. If it is necessary to rollback to a previous state of the database, then transaction time support is called for.

TEMPORAL RELATIONS

A temporal relation is of the form:
$R(A_1,A_2\ldots,A_n|t^b)$
It consists of a number of attribute values associated with a bitemporal timestamp value t^b. The bitemporal timestamp value t^b is represented by the ordered pair (c^t,c^v) with c^t representing the transaction time and c^v the valid time.
An example of a temporal relation would be the military officers' temporal relation presented in figure

3. The valid time interval determines when the information stored in the tuple was valid. From "1953/3" till "1985/8" the military officer "Johnson" was a Major General in the army and his employment started on the year 1953.

ID	Name	Empl. Date	Rank	ValidTime
100	Johnson	1953	Major General	[1953/3-1985/8]
101	Miles	1983	Lieutenant General	[1983/7-∞]

Figure 3. A sample data of the military officers' temporal relation.

4. TEMPORAL MULTILEVEL SECURE RELATION

A multilevel secure temporal relation is of the form

$$R(A_1,C_1,A_2,C_2,\ldots,A_n,C_n,VT,C_{vt},TC)$$

where A_i is a data attribute over domain D_i, C_i is a classification attribute for A_i, VT is the valid time attribute, C_{vt} is a classification attribute T, and TC is the tuple-class attribute. The domain of C_i is specified by a set $\{L_i, \ldots, H_i\}$, which enumerates the allowed values for access classes, ranging from the greatest lower bound (glb) L_i to the least upper bound (lub) H_i. The domain of TC is the set {lub{Li; i=1,…,n},…, lub{Hi: i= 1,…,n}}, and the domain of C_{vt} is the set {lub{Li; i=1,…,n},…, lub{Hi: i= 1,…,n}.

4.1 UPDATE OPERATIONS

The operations on a relational database can be categorized into two main categories: retrievals and updates.

The update operations can be divided into three types of operations: **Insert** which is used to insert a new tuple or tuples in a relation, **Delete** which is used to delete tuples, and **Modify** which is used to change the values of some attributes.

Changes to temporal databases are viewed as additions to the information stored in the database. Since in a temporal database no data is ever deleted, meaning once data is inserted into the database it will not be deleted at any other time, but rather a new tuple reflecting the changes to the data is inserted with a new timestamp value. The same will apply to temporal multilevel secure relational databases. So we only need to worry about insert and modify operations in a temporal multilevel secure database.

For multilevel secure databases, insert and update operations are carried in a very much similar way as they are carried in classical databases except for certain updates that do not simply involve the overwriting of data since this would lead to a failure in ensuring the secrecy of data.

In this section, we show by examples how update operations take place in temporal multilevel secure databases. Whenever we need to do an update operation, we need to ensure not to violate the integrity constraints specified on the database.

Consider the following example of the multilevel relation *military officers* as depicted in Figure 4. The figure shows a set of sample values for the *military officers'* relation.

ID	Name	Empl_Date	Rank	TC
100 U	Johnson U	1953 U	Major General U	U
101 S	Miles S	1934 S	Marshal S	S

Figure 4. A set of sample values for the military officers' relation.

Access clearances are assigned to individual data elements of a relation. Subjects having different clearances see different versions of the military officers' relation. A U-User having a clearance at the access class U will see a version of the military officers' relation that includes only the data that were assigned an access class U. While an S-User will be able to see a version of the military officers table that will include both the data that were assigned an access class U and an access class S.

In order to be able to record time varying data into our database we need to extend the military officers' relation by adding the temporal attribute ValidTime. It is an interval that we use to determine when the data inserted into the tuple, was, is or will be valid. The U-user and the S-User version of the employees table after adding the temporal attribute ValidTime are shown in Figure 5 and Figure 6.

ID	Name	Empl_Date	Rank	ValidTime	TC
100 U	Johnson U	1953 U	Major General U	[1953/3 -∞] U	U

Military_Officers$_u$

Figure 5. The U-User version of the *military officers'* relation after adding the temporal attribute ValidTime.

No	Name	Empl_Date	Rank	ValidTime	TC
100 U	Johnson U	1953 U	Major General U	[1953/3-∞] U	U
101 S	Miles S	1934 S	Inspector General S	[1985/7-∞] S	S

Military_Officers$_S$

Figure 6. The S-User version of the *military officers'* relation after adding the temporal attribute ValidTime.

Let us go back to our main objective of this section, which is how to handle update operations in a temporal multilevel secure database!

Let us assume that on April 1981 a U-user wants to update the rank of the military officer "Johnson" from "Major General" to "Lieutenant General". Figure 7 shows the U_User version of the *military officers* table, and figure 8 shows the S-User version after this update.

No	Name	Empl_Date	Rank	ValidTime	TC
100 U	Johnson U	1953 U	Major General U	[1953/3-1981/4] U	U
100 U	Johnson U	1953 U	Lieutenant General U	[1981/4-∞] U	U

Military_Officers$_u$

Figure 7. The changes to the U-User version of the military officers after updating the rank of military officer "Johnson" by a U-User.

ID	Name	Empl_Date	Rank	ValidTime	TC
100 U	Johnson U	1953 U	General U	[1953/3-1981/4] U	U
100 U	Johnson U	1953 U	Lieutenant General U	[1981/4-∞] U	U
101 S	Miles S	1934 S	Inspector General S	[1985/7-∞] S	S

Military_Officers$_S$

Figure 8. The changes to the S-User version of the military officers after updating the rank of military officer "Johnson" by a U-User.

As a result to this update a whole new tuple had to be inserted. This new tuple is inserted at the U class. The valid time for the old tuple of the military officer "Johnson" is updated to reflect the time in history when the rank of officer "Johnson" was "Major General". From the date April 1981, the rank of the officer "Johnson" changed to "Lieutenant General".

As we can see from the example, an update performed by a user with an X clearance on a tuple with an access privilege X is dealt with the same way we deal with an update operation in the temporal database model with the addition that the new inserted tuple will also have an access privilege X.

Let us consider another example, in which we deal with the case where a higher level user tries to update a tuple that has lower level access privilege. Going back to our military officers' example, if a new tuple was inserted by a U-user to the *military officers* table then it is possible for any attribute within that tuple to be updated by an S-User. Assume that on January 1, 1997, a user with an S clearance gives the two officers "Johnson" and "Miles" a new higher rank. Since in temporal database whenever we are updating we do not actually update the value, but we rather insert a new tuple with the same values for all the attributes, except for the attribute that is to be updated and issued a new timestamp value; this means that the S-user would have to insert a new tuple. This tuple would be inserted at the S-level since an S-user is performing the operation (see figures 9 and 10). But this would create a problem, because we would have two tuples with the same apparent key with overlapping time timestamps.

ID	Name	Empl _Date	Rank	ValidTime	TC
100 U	Johnson U	1953 U	Major General U	[1953/3-1981/4] U	U
100 U	Johnson U	1953 U	Lieutenant General U	[1981/4-∞] U	U

Figure 9. Military_Officers$_u$.

ID	Name	Empl _Date	Rank	ValidTime	TC
100 U	Johnson U	1953 U	Major General U	[1953/3-1981/4] U	U
100 U	Johnson U	1953 U	Lieutenant General U	[1981/4-∞] U	U
100 S	Johnson S	1953 S	Inspector General S	[1997/1-∞] S	S
101 S	Miles S	1934 S	Inspector General S	[1985/7-1997/1] S	S
101 S	Miles S	1934 S	Marshal S	[1997/1-∞] S	S

Figure 10. Military_Officers$_S$.

This would result in a temporary inconsistency in the database that needs to be resolved. The inconsistency can be resolved as follows: The S-user logs on at the U-level and insert a new tuple with a nullified rank value that happens to have the same timestamp of the tuple inserted at the S-level. Figures 11 and 12 show what the relation would look like.

ID	Name	Empl _Date	Rank	ValidTime	TC
100 U	Johnson U	1953 U	Major General U	[1953/3-1981/4] U	U
100 U	Johnson U	1953 U	Lieutenant General U	[1981/4-1997/1] U	U
100 U	Johnson U	1953 U	Null U	[1997/1-∞] U	U

Figure 11. Military_Officers$_u$.

ID	Name	Empl _Date	Rank	ValidTime	TC
100 U	Johnson U	1953 U	Major General U	[1953/3-1981/4] U	U
100 U	Johnson U	1953 U	Lieutenant General U	[1981/4-∞] U	U
100 U	Johnson U	1953 U	Null U	[1997/1-∞] U	U
100 S	Johnson S	1953 S	Inspector General S	[1997/1-∞] S	S
101 S	Miles S	1934 S	Inspector General S	[1985/7-1997/1] S	S
101 S	Miles S	1934 S	Marshal S	[1997/1-∞] S	S

Figure 12. Military_Officers$_S$.

This scheme will not create a downward signaling channel from one subject to another since the nullification at the U-level is being done by a U-subject. Someone might say that there is a downward signaling channel with a human in the loop. The human, however, is trusted not to let the channel be exercised without good cause.

The coexistence of the tuple (100, Johnson, 1954, Inspector General, [1997/1-∞]) and the tuple (100, Johnson, 1954, null,[1997/1-∞]) in Military Officers$_S$, two tuples with the same primary key, is what is referred to as polyinstantiation [4]. Here, there is no threat of entity or attribute polyinstantiation, because at any time the attribute value is updated this means that a new tuple would need to be inserted with the same primary key, same time timestamps, but with different value for the attribute at each level, the value of the attribute would appear null at the lowest level, if this attribute was updated by a higher level user.

Another problem that the coexistence of these two tuples might create is that they both have the same time timestamps. In temporal databases at any given instance of time each military officer is supposed to have only one rank. This problem is referred to as the contradiction problem [9]. Since the military officer 100 is shown to have a rank of both null and "Inspector General" from the date January 1997 and up to this date.

Let us take another example to clarify the problem; we take a stock database containing the *stkwh* table showing the quantity available in a certain warehouse of a certain product at a specific period in time. The relation scheme for the relation stkwh is presented below:

STKWH(wrh_no, prod_no, qty_avlb)

Figures 13 and 14 show a set of sample values for the *STKWH* relation.

Wrh_No	Prod_No	Qty_Avlb	ValidTime	TC
100 U	1014 U	2000 U	[1998/3-1998/4] U	U
100 U	1014 U	0 U	[1998/4-∞] U	
100 U	1020 U	500 U	[2004/1-∞] U	U

STKWH_u

Figure 13. A set of Sample Values for the *STKWH* relation.

Wrh_No	Prod_No	Qty_Avlb	ValidTime	TC
100 U	1014 U	2000 U	[1998/3-1998/4] U	U
100 U	1014 U	0 U	[1998/4-∞] U	U
100 U	1020 U	Null U	[2003/9-∞] U	U
100 U	1020 U	500 U	[2004/1-∞] U	U
100 S	1020 S	1000 S	[2003/9-∞] S	S

STKWH_s

Figure 14. A set of Sample Values for the *STKWH* relation.

Let us assume that an S-User wants to know the quantity of the product 1020 currently available in the warehouse 100. This is where the contradiction problem appears - we have two tuples with the same primary key showing the quantity available in the warehouse 100 of item 1020. Which one of the two quantities is the true available quantity 500 or 1000? We only have this problem for high level users. Our next step is to try solving this problem.

4.2 INTEGRITY CONSTRAINTS

In this section, we are concerned with identifying the integrity constraints that must hold on a temporal multilevel secure database, in order to ensure that all the tuples in the database are meaningful. Since we are talking about temporal multilevel secure databases, we will try to identify the integrity constraints we need to specify on temporal multilevel secure databases by combining the integrity constraints specified on both the temporal and multilevel secure databases.

In multilevel temporal databases, we store different database states, and users with different clearances see different versions of these database states. These different versions must be kept coherent and consistent, without introducing any downward signaling channels. All the tuples in the database must be meaningful, so we should not have redundancy, circumlocution or contradiction problems. To be able to meet all of these requirements we need to specify some constraints on temporal multilevel secure databases. These constraints must be a combination of the integrity constraints of temporal databases along with those of multilevel secure databases.

ENTITY INTEGRITY: Let AK be the apparent key of R, and let VT be the valid time of R, A temporal multilevel relation R satisfies entity integrity if and only if for all instances R_c of R and $t \in R_c$:

1. $A_i \in AK \Rightarrow [A_i] \neq \text{null}$
2. $[VT_i] \neq \text{null}$
3. $A_i, A_j \in AK \Rightarrow t[C_i] = t[C_j] = t[C_{VT}]$ (where C_{VT} is the classification of the valid time)
4. $A_i \notin AK$ and $A_i <> VT_i \Rightarrow t[C_i] \geq t[C_{AK}]$ (where C_{AK} is the classification of the apparent key)

In multilevel secure relational model, we have three requirements in the entity integrity constraint to specify that no part of the primary key can have a null. Extending these requirements to temporal multilevel secure relational model we obtained the above mentioned four requirements. The first requirement ensures that no attribute of a primary key of a base relation may be null. The second requirement specifies that the valid time value can never be null. The third requirement ensures that all the attributes of a primary key of a base relation must have the same access class, and not only the valid time access class must also have the same access class as these attributes. The fourth requirement says that the access class of all non-key attributes (the valid time is not included) in a tuple dominates the access class of the primary key.

NULL INTEGRITY: A multilevel temporal relation R satisfies null integrity if and only if for each instance R_c of R both of the following conditions are true:

1. For all $t \in R_c$, $t[A_i] = \text{null} \Rightarrow t[C_i] = t[C_{AK}]$;

2. Let us say that tuple t subsumes tuple s if for every attribute A_i, either
    a. $t[VT_i]$ overlaps $s[VT_i]$ and $t[A_i] \neq s[A_i]$
    or
    b. $t[A_i] = s[A_i]$ and $t[VT_i]$ merges $s[VT_i]$

The first requirement means that attributes that have null values have an access class that is equal to the access class of the primary key. The second requirement states that R_c does not contain two distinct tuples with different non-key attributes values and the valid time of one overlaps the valid time of another, or two distinct tuples with identical value for all the attributes and the valid time of one merges the valid time of another. Having such tuples will lead to a problem similar to one of the problems we had in temporal relational model, the redundancy, circumlocution or contradiction problem. That is why we need to prevent the existence of such tuples either by combining the tuples that have a redundancy or circumlocution problem or by preventing the existence of tuples that would cause contradiction (note we are talking about the attributes that would an access class similar to that of the primary key and therefore similar to that of the valid time).

INTERINSTANCE INTEGRITY: R satisfies interinstance integrity if and only if for all $c' \leq c$ we have $R_{c'} = \sigma (R_c, c')$, where the filter function σ produces the c' – instance $R_{c'}$, from R_c as follows:

1. For every tuple $t \in R_c$ such that $t[C_{AK}] = c'$, there is a tuple $t' \in R_{c'}$ with $t'[AK,C_{AK}]$ and for $A_i \notin AK$

$t'[A_i,C_i] = \{$	$t[A_i,C_i]$ if $t[C_i] \leq c'$
	<null,$t[C_{AK}]$> otherwise

2. There are no tuples in Rc' other than those derived by the above rule.
3. If at any given time the end result contained two tuples that have the same apparent primary key value (the non valid time attributes of the primary key) but differ on the values of their non-key attributes then their valid time values i1 and i2 must be such that i1 overlaps i2 is false.
4. If at any time the end result contained two distinct tuples that are identical except for their valid time values i1 and i2, then i1 merges i2 must be false.

In this constraint, the filter function is used to map the multilevel temporal relation to different instances, one for each access class, so as to give the user the ability to see only the historical data for which he/she is cleared. The resulting obtained instance will be similar in a way to a temporal database. In addition, we must ensure in the end result to combine the tuples that cause redundancy or circumlocution, and not to have two tuples that lead to a contradiction.

POLYINSTANTIATION INTEGRITY: In temporal multilevel secure databases, we may have several tuples with the same primary key but with different values for the non-key attributes. Not only this, even at the same access level we will have more than one tuple with the same primary key but with different valid times. As previously mentioned in multilevel secure databases we can not prevent a low user from inserting a tuple with the same primary key as a previously inserted high level tuple or we might create some downward signaling channel that will violate the secrecy of data. At the same time we can not prevent a user at the same access level from inserting a tuple with the same primary key as an old existing tuple at the same access level but with different valid time. We can either refuse such an insertion or override existing data. Refusing to insert this tuple, or overriding existing data for either any of the two previously mentioned reasons will cause a downward signaling channel, the loss of secret information, and the destruction of historical data. We have no choice but to keep all the tuples without violating the foundations of relational databases. That is why we need to declare the access class, and the valid time to be part of the primary key. So we need to specify the following key constraint:

R satisfies the key integrity if and only if for every R_c we have for all A_i: AK, VT, C_{AK}, C_{VT}, $C_i \rightarrow A_i$, this means that the user specified primary key AK in conjunction with the valid time, the classification attributes C_{AK}, the classification attribute C_{VT}, and C_i, functionally determines the values of Ai attribute.

5. CONCLUSION

In this paper we introduced a new database model - the multilevel secure temporal relational data model. We discussed update operations and how they take place in our model. We also specified the integrity constraints needed in a temporal multilevel secure database in order to ensure that all the data inserted in the database is consistent, meaningful, historical and secure.

REFERENCES

[1] Chen, F., and Sandhu, R. *The Multilevel Relational Data Model*. ACM Transactions on Information and System Security, Vol. 1, No. 1, pp. 93-132, 1998.

[2] Date, C., Darwen, H. and Lorentzos, N. *Temporal Data and the Relational Model*. Morgan Kaufmann. San Francisco, 2003.

[3] Jajodia, S., Sandhu, R., and Blaustein, B. *Solutions to the Polyinstantiation Problem*. Information Security, pp. 460-492, 1994.

[4] Jajodia, S., Sandhu, R., and Blaustein, B. *Towards a Multilevel Secure Relational Data Model.* Information Security, pp. 493-529, 1994.

[5] Ozsoyoglu, G., and Snodgrass R. *Temporal and Real-Time Databases: A Survey*. IEEE Transactions on Knowledge and Data Engineering. Vol. 7, No. 4, August 1995.

[6] Pissinou, N., and Makki, K. *Towards a Framework for Integrating Multilevel Secure Models and Temporal Data Models*. Proceedings of the Third International Conference on Information and Knowledge Management. Gaithersburg, USA, pp. 280-287. 1994.

[7] Smith, K. and Winslett, M. *Entity Modeling in the MLS Relational Model*. Proceedings of the International Conference on Very Large Databases, Los Alamitos, USA, pp. 199-210. 1992.

[8] Shasi, K, *Applicability of Temporal Data Models to Query Multilevel Security Databases: A Case Study*. Temporal Databases – Research and Practice. LNCS 1399, pp. 238-256, 1998.

[9] Tansel, A., Clifford, J., Jajodia, S., Segev, A., and Snodgrass, R. *Temporal Databases: Theory, Design and Implementation*. The Benjamin Cummings Publishing Company, Redwood City, USA, 1993.

Ramzi A. Haraty is an associate professor and the chairman of the Division of Computer Science and Mathematics at the Lebanese American University in Beirut, Lebanon. He is also the Chief Financial Officer of the Arab Computer Society. He received his B.S. and M.S. degrees in Computer Science from Minnesota State University - Mankato, Minnesota, and his Ph.D. in Computer Science from North Dakota State University - Fargo, North Dakota. His research interests include database management systems, artificial intelligence, and multilevel secure systems engineering. He has well over 80 journal and conference paper publications. He is a member of Association of Computing Machinery, Arab Computer Society and International Society for Computers and Their Applications.

Natalie Bekaii received her B.S. and M.S. degrees in Computer Science at the Lebanese American University – Beirut, Lebanon. Her research interests include temporal databases and multilevel security.