In Spooner, SA Demurjian and JE Dobson (eds), *Database Security IX: Status and Prospects*, 27–42, Chapman & Hall, 1996

Self-protecting Objects in a Secure Federated Database

Martin S Olivier
Department of Computer Science, Rand Afrikaans University
PO Box 524, Auckland Park, Johannesburg, 2006 South Africa
Email: molivier@rkw.rau.ac.za

Abstract

This paper presents an implementation strategy for a secure federated database. A federated database is a distributed database with a relatively high degree of site autonomy. The proposed implementation strategy assumes that a federal security policy specifies the security aspects that apply to all sites of the federation. Each site is then able to extend the security policy that applies to data owned by it with its own site security policy. The site security policies are guaranteed to be enforced even if an object of one site is relocated to another site.

Keywords

Keyword codes: C.2.4, H.2.5, K.6.5
Keywords: Distributed Systems, Heterogeneous Databases, Security and Protection

1 INTRODUCTION

An object is an encapsulated unit that combines (traditional) code and data into a single entity. In principle, it is possible to not only include code that performs the expected data manipulation, but also the code that provides the required security for the object. This concept of *self-protecting objects* has been used by Olivier and Von Solms (1992) and Olivier (1994). The concept has the greatest potential benefit in a distributed database where an object often has to be relocated from one site to another—the security code of the object can then be relocated with it and the object will remain as protected at its new location as it has been at its original location. This is especially true if the concerned database is a federated database—that is a database where sites are essentially autonomous (but do cooperate) (Özsu and Valduriez, 1991).

According to Ceri and Pelagatti *"site autonomy is achieved when each site is able both to control accesses from other sites to its own data and to manipulate its data without being conditioned by any other site"* (Ceri and Pelagatti, 1985:323). By implication, each site of the federation should be free to select its own security policy and this policy should be respected by other sites in the federation. This can be accomplished by including the

protecting code in the object—in other words, to let the objects 'protect themselves' in accordance with the security policy of their site.

However, it is necessary to show that

1. It is possible to construct a system where the self-protecting code can be attached to the object by the security system—it cannot be the responsibility of application programmers to include security code when developing the system itself; and
2. Such protection can be accomplished by compact, clear, straightforward code in order to be trusted.

The purpose of this paper is to present an implementation strategy for self-protecting objects that may be used in a federated database.

The paper describes an implementation strategy that influences the design of the DBMS of each site of the federated database. This strategy therefore implies that the DBMSs of the sites have to be written especially for our model—it is not possible to add the described security functionality to an existing DBMS in the same way that an integrity lock can be added (see Denning, 1988:12, for example). Often the problem posed by federated databases is precisely the fact that existing DBMSs have to be integrated, and much of the current research into security of such databases therefore focusses on integration of the existing systems—see Gong and Qian (1994) and Jonscher and Dittrich (1994) for example.

However, there are cases where federated databases are designed from the top down (rather than integrated from the bottom up). In fact, Özsu and Valduriez (1991:81,89) use the term *federated database* for systems that are usually designed from the top down and the term *multidatabase* for systems that are integrated from existing databases. Other authors (for example Jonscher and Dittrich, 1994) prefer to refer to the first category as *tightly coupled federated databases*, and refer to systems that are integrated from existing centralised databases as *loosely coupled federated databases*. The work reported in this paper is clearly applicable to databases in the first category.

Even though a (tightly coupled) federated database may be constructed from the top down, it is still possible that different security policies may apply at different sites. This may, for example, happen when a federated database combines DBMSs that belong to different organisations and the organisations prefer to use different security policies. In fact, the same database may be part of two separate federations which almost makes it impossible for the federations to prescribe a security policy for the site. Phrased in terms of security policies, where the described implementation strategy is used, a federal security policy will describe a baseline of security that exists for all sites of the federation. Each site may then have its own site security policy that extends the federal policy as desired by the site. An (implicit) entry in the federal security policy is that all sites will honour another site's policy whenever data from that site is accessed away from that site.

The concept of a federated security policy has been used by Thuraisingham (1994). However, in contrast to our view given in the previous paragraph, Thuraisingham (1994) defines the federated security policy as the combination of all the local security policies. We prefer to see the *federal security policy* as the security requirements that apply to all sites, before local extensions from any particular site are considered. Although not used in this paper, we also prefer to refer to the security policy that applies to the members of the federation after all local security policies have been incorporated, as the *effective*

federal security policy. The *effective federal security policy* does not need to be expressed explicitly.

We will use the acronym SPO *(Self-protecting object model)* for the implementation strategy described in this paper.

After summarising the relevant background in section 2, section 3 will describe the components of a federated database based on SPO. This is followed in section 4 by some remarks about object relocation and replication in the federation. Section 5 argues that it is possible to build a database that can be trusted using SPO. After this, section 6 considers the methods used to protect entities. An introduction to a formalisation of SPO follows. Finally, the conclusion considers the efficiency of SPO, as well as future research.

2 BACKGROUND

SPO is based on the object-oriented paradigm: All entities in the database are objects. An object combines code (methods) and data (in instance variables) into a single, encapsulated entity. In order to use an object, a message is sent to the object; a corresponding method of the object handles the message in an appropriate way. Objects are 'instantiated' from classes: a class is a 'template' that describes an object. In general, any number of objects can be instantiated from the same class. When a class is defined, it can be based on previously defined classes; in such a case the new (sub-) class inherits method and variable definitions from the previously existing (super-) class. An object-oriented database is a database that is based on object-oriented concepts. See Wegner (1990) for a discussion of object-oriented concepts and Kim (1991) for a discussion of object-oriented database concepts.

A federated database is a distributed database where the sites that form the distributed database have a high degree of site autonomy—that is, management of the distributed database is delegated to the sites as far as possible. See Özsu and Valduriez (1991) for a discussion of distributed databases and Özsu and Valduriez (1991:81,89) and Sheth and Larson (1990) for a discussion of federated databases in particular.

Database security ensures confidentiality, integrity and availability of data maintained in a database. Many models for secure databases have been proposed—see Castano *et al* (1995) and Denning (1988) for a comprehensive treatment of database security and Olivier and Von Solms (1994) and Rabitti *et al* (1991) for a discussion of security in object-oriented databases.

SPO is intended for use in secure object-oriented federated databases. The object-oriented paradigm is used because the concept of self-protecting objects fits in well with the object-oriented paradigm. Further, SPO forms part of a larger research project that focusses specifically on security in object-oriented databases.

Security in federated databases has already received attention in the literature: Thuraisingham (1994) highlights the (many) issues that must be considered when designing a secure federated database. Bull, Gong and Sollins (1992) argue that security in such a system should be approached from the 'server' (and not the user). Gong and Qian (1994) show that interoperation of systems can cause unintended (indirect) access to information and prove that elimination of such unintended access for a simplified case is NP complete. One solution is to achieve secure global interoperation "incrementally by composing secure local interoperation." Pernul (1993) studies the policy and mechanisms that can be used

on the federal layer of a federated database, given heterogeneous security mechanisms at the various sites. Idris *et al* (1994) consider the integration of secrecy when schemas are integrated to form a federal schema. Jajodia *et al* (1994) show that not all combinations of concurrency control algorithms in distributed databases ensure global database consistency. Olivier (1994) assumes homogenous security mechanisms and essentially homogeneous security policies at the various sites, but considers the problem that occurs if local autonomy leads to different authorisation policies at different sites. The proposed model ensures that data owned by one site cannot flow to a user at another site if the owning site does not agree with the authorisation policy of the other site. Jonscher and Dittrich (1994) investigate the required security functionality of the federated layer of a tightly coupled federation for discretionary access control. The interaction of this layer with the security systems of the individual sites is also considered.

The current work differs from those mentioned earlier since it describes an implementation strategy for a secure federated database that allows a high degree of local (autonomous) determination of the site security policy and ensures that the, potentially different, policies are all enforced throughout the federation.

SPO is based on a common core of security code: this code (identical or equivalent) executes at all sites of the federation. This core may implement normal database security functions on a federation-wide basis. However, the most important task of this code is to ensure that the security that is implemented by any other site, is supported on all other sites. Obviously, this core needs to be trusted by all members of the federation. In addition to the common core, individual sites may implement extension security code to address their particular security needs. Such an extension only needs to be trusted by the site that implements it.

When a new site wants to join the federation, it will first implement the common core of security code to the satisfaction of other federation members; after this the new site will implement its security extensions according to its own security policy (to protect its own objects). At this point the new site may become a member of the (secure) federation.

3 THE SPO ARCHITECTURE

This section describes the components of SPO and their relation to one another.

The common security core mentioned in the previous section will be referred to by the acronym TCC *(Trusted Common Core)*. The TCC executes on all sites of the federation and is responsible for implementing the federal security policy (even if that policy only states that site security policies should be honoured by other sites of the federation)—in particular, the TCCs are responsible for enforcing access control and information flow restrictions. For simplicity (and to aid mutual trust between federation members) the TCC running on each site may be identical to the TCCs on all other sites. However, if so required, the TCCs can be implemented differently, as long as they are equivalent. In particular, there is no reason that the TCCs cannot be implemented on dissimilar computers that participate the same federation.

Each node can add its own security code 'on top' of the TCC. Such extensions are grouped into two categories: In some cases the security code will be included (encapsulated) in the object and may accompany the object if the object is relocated to a new site. These extensions will be referred to as *Trusted Extensions* (TEs). The other category

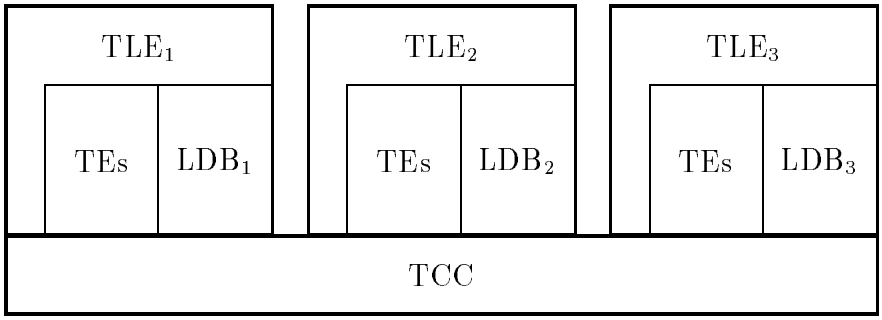

Figure 1 The components of SPO

of extensions will not be used to protect individual objects, but rather to provide the support required by the TEs. These other extensions will be referred to as *Trusted Local Extensions* (TLEs). This name stems from the fact that TLEs will always remain at the site that defines them. Together, the TLE and TEs of a site implement the site security policy for that site.

The components of SPO are depicted in figure 1—in this case a federation consisting of three sites. Each site has its own TLE. Each site has TE code, portions of which may have been obtained from other sites along with objects obtained from those sites. Each site contains a TCC (which is identical in operation to TCCs at all the other sites). Since the TCCs are equivalent and can communicate, they are depicted as a single layer in this figure. The individual local databases are marked LDB_1, LDB_2 and LDB_3; these local databases may contain objects owned by that site as well as objects relocated from other sites, including replicated objects.

As an example, assume that a library is about to form part of a federation. Assume that this library will form a site known as LIBRARY. The components of the LIBRARY site will be discussed under the headings *Local security policy*, *Operation of TEs*, *Operation of the TCC* and *Operation of the TLE*.

3.1 Local security policy

We will assume that LIBRARY is willing to share information about publications on its shelves freely with other members of the federation. However, updates to information are strictly controlled: lending out details (name of borrower, date due back) can only be changed at the lending desk of the LIBRARY itself, while records (objects) about books and other material can only be inserted and deleted by members of the acquisition group of the library. Similarly, information about the contents of a book can only be changed by a subject librarian.

The application objects (about books, borrowers, and so on) will be implemented by programmers in the usual way.

3.2 Operation of TEs

The security policy has been outlined above. The SSO (system security officer) will next identify the required TE methods to protect the objects: AbortIfOriginNotFrontDesk, AbortIfNotAcquistionGroup and AbortIfNotSubjectLibrarian. It is possible that such methods may already exist; if they do not, they will have to be written by the programmers who normally write trusted code and then be verified according to the security policy of the site. (Some alternatives to actually writing trusted code for each item to be protected will be discussed in section 6.) As stated earlier, these TE methods only need to be trusted by the concerned library—other members of the federation do not even have to know about the methods' existence at this point.

The SSO can now associate the TE methods with the application methods: for example, the IssueBookToBorrower method can be protected with the AbortIfOriginNotFrontDesk method. Once the objects have been thus protected, they can be entered at the local site (and hence into the federated database). The federal security policy may require further steps from the local site; for the sake of this example we will assume that it does not.

3.3 Operation of the TCC

Suppose that a book object of Library currently resides at some site S_1. Further, assume that copies of the TEs that protect this object also reside at site S_1. If a message (say IssueBookToBorrower) is now sent to this object, the TCC will be involved before the corresponding method is activated. In particular, the TCC at site S_1 will maintain a list of objects currently residing at S_1 and of TE methods that are associated with those objects and their various facets (such as methods and attributes). Therefore, before the TCC at S_1 will allow the IssueBookToBorrower method to be activated, the TCC will execute the AbortIfOriginNotFrontDesk TE method, which will perform access checks according to the site security policy of Library.

Since the TCC at S_1 is trusted by Library, Library can rest assured that the appropriate TE methods will always be invoked, that those TE methods will be properly executed and that the decision of the TE method will be enforced by the TCC.

Since the TCC has to maintain a list of objects residing at its site as well as a list of TE methods used to protect those objects, it is obvious that such information has to be supplied by any TCC from which an object is relocated to the TCC at the site to which an object is relocated. This will be discussed in more detail in section 4.

3.4 Operation of the TLE

As stated earlier, the TCC will ensure that the site security policy will be enforced, even if an object is used at another site. However, one cannot assume that the TCC on its own will provide all the necessary support for the site security policy: In the example above, the federal security policy (and hence, the TCC) may say nothing about groups, even though the site policy mentions an acquisition group and a subject librarian group. This is solved as follows: The TLE at the site that wants to use group information will maintain such information at that site; the TE that protects the concerned entity and needs the group information, will address a question to the TLE about the group membership of the

message sender. The only function of the TCC in this regard is that it provides a secure communication channel between the TE and the TLE. Similarly, if information regarding the origin of a message is required for security purposes, this can be collected by the TLE that requires it.

Obviously, the approach described in the previous paragraph potentially limits cooperation. Since other sites may not be interested in the origin of the message, they may not collect information about the origin. If a user at another site were to try to modify lending out details of an object belonging to LIBRARY, the TE protecting that object will request the origin of the message from the TLE at LIBRARY. If the message did not originate from LIBRARY then the TLE at LIBRARY will not have details about the origin; if such details are not available, execution will be aborted just as in the case where the message originated from a known, but unacceptable terminal.

In the case described in the previous paragraph the described behaviour is probably the desired behaviour—users at other institutions should be prevented from changing the lending out details (according to LIBRARY's security policy). However, when joining the federation, LIBRARY may determine that a group of users at a site LIBRARY2 are also skilled subject librarians and that it may be mutually beneficial if the subject librarians at the two sites can modify information about the contents of books owned by either of the two sites—assume that both sites have a SUBJECTLIBRARIAN group and LIBRARY wants to allow users in the SUBJECTLIBRARIAN group of LIBRARY2 to be also able to modify content information about books. Now, whenever a message is sent to a book object owned by LIBRARY to change the information about a book's contents, the appropriate TE will contact the TLE at LIBRARY to determine whether the sender of the message is a member of SUBJECTLIBRARIAN. However, if the sender's home site is LIBRARY2 then the TLE at LIBRARY simply will not know. This problem has two potential solutions. The first is to include the notion of a SUBJECTLIBRARIAN group in the federal security policy. The other solution is for LIBRARY to bilaterally agree with LIBRARY2 to exchange information about users in the SUBJECTLIBRARIAN group. The latter possibility can be implemented in a number of ways: As an example, if the TLE at LIBRARY is asked whether user X is a member of SUBJECTLIBRARIAN, and LIBRARY determines that it (LIBRARY) is not the home site of user X, it can pass the question on to LIBRARY2 (via a secure communication channel); if LIBRARY2 answers in the affirmative, the request is allowed to proceed; if LIBRARY2 denies it or does not know, the request is aborted. The solution selected depends on a number of factors, amongst them the proportion of the federation that will benefit from a federal implementation of the concept and the time available. It is envisaged that most cases like this will begin as a bilateral agreement, expand to include more parties and, if it proves to be generally usable, to be included in the federal policy at a revision point of the federal policy.

3.5 Summary of architecture

The TCC implements the federal security policy (including ensuring that the local security policies are respected globally). Since the TCC applies to the entire federation the TCCs running at the various sites should be equivalent to one another; all TCCs should be trusted by all members of the federation. The TLE implements the remaining security functionality required according to the the security policy for a particular site. The TE contains code tightly associated with the objects to be protected. TE code may be relo-

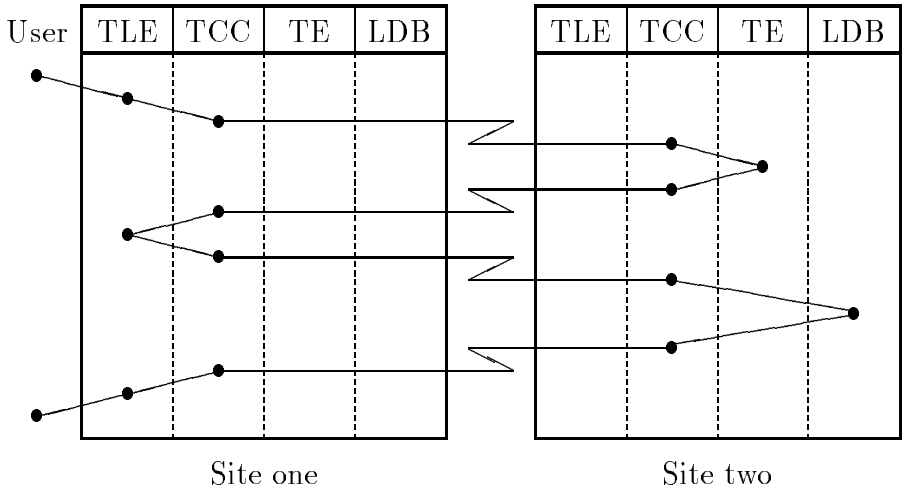

Figure 2 Example of flow of control in a two site SPO federated database

cated with objects; TLE code will always remain at its home site. The TLE may include functions such as local user authentication and authorisation, unless that is deemed a part of the federal security policy. The TLE will also provide information about users and messages as requested by a TE. The information that can be supplied by a TLE about its users or its messages depends on what should be available according to the security policy of the TLE's site.

Figure 2 illustrates the flow of information in an SPO federation. A message is sent from a user at the first site to an object on a second site. The message enters the system at the TLE at that site. From there it is passed to the TCC. Since the target object is on another site, the message is now passed to the TCC at this site. Here the TCC performs access checks, including execution of the appropriate TE. Assume that the TE requires further information about the message to complete its checks. The TE therefore informs the TCC what additional information is required and what valid values would be. The TCC now requests this information from the first site, since the message originated there. At the first site, the TCC obtains this information from the TLE and sends it to the requesting TCC. This TCC performs the final checks and then delivers the message to the target object in the LDB at the second site. The reply is now sent to the user via the TCC at the second site and the TCC and TLE at the first site.

4 RELOCATION AND REPLICATION

In a federated database the need exists to relocate objects to new sites and to replicate objects at more than one site for reasons of efficiency. See Lee, Wang and Chang (1994) for a discussion of the reasons and problems behind object relocation in an object-oriented system, as well as an approach that enables object movement to be accomplished across heterogeneous systems. Although they do not specifically address databases, their remarks do apply to object-oriented databases. They do not address security issues. In SPO we

assume that the data models used at the various sites are homogeneous to focus attention on the issue of heterogeneous security policies; heterogeneous data models will be addressed in future research.

In order to ensure that security is not compromised when relocating an object in SPO, such relocation is done with the aid of the TCC, and TE methods may accompany the object to protect it at the new site.

To illustrate relocation and replication, assume that a number of hospitals and laboratories share a federated database. When a test is required at a hospital, the request is sent to the relevant laboratory to perform it. After performing the test, the laboratory will create a test report in the database. Since it is imperative that only the laboratory personnel be able to insert test results in this report, the report object will be protected accordingly. However, once the report has been created it will probably only be used by staff at the concerned hospital. It therefore makes sense to relocate the report to the hospital site of the federated database—with its original (laboratory site) protection.

In the same federation the various laboratories may publish lists of the tests that they perform (possibly including cost information). Since such lists may be used frequently by the hospitals (possibly every time a test is requested), it may be beneficial if the lists are replicated at all the hospitals. Again the lists may need protection—for example, it may be necessary to ensure that cost information can only be changed by a particular manager at the concerned laboratory. Also here the required protection is added by the concerned laboratory; however, such protection should hold for all replicated copies of the list.

True replication may not always be necessary: the example of the previous paragraph may also use *snapshots* (Date, 1985:306–9): Here the copies are sent to the sharing sites on a regular basis—for lists of tests and corresponding costs this may possibly be done on a monthly basis. Changes to the master copy are then not reflected in the snapshots immediately when made, but only at the point when a new snapshot is sent to the site. However, even if an object is replicated in the form of snapshots, the snapshots should still be protected appropriately.

With these examples in mind we can now discuss the issues surrounding relocation and replication in SPO.

Whenever an object is relocated (or replicated) it will still be protected by the TE methods that protected it at its original site. Although it is possible for the TE methods to remain at the original site when the object is relocated, it may reduce communication overhead if the TE methods are relocated with the object (since each access to a facet of the object will invoke the TE method that protects that facet). The protocol to accomplish this will operate as follows: Firstly, the TCC at the sending site will determine which TE methods are used to protect the object to be relocated at its current site. Copies of these methods will then be sent to the TCC at the receiving site. After this, the concerned object will be relocated to its new site and the TCC at this site will then use the copies of the TE methods to protect the relocated object. Since the TCC is trusted by all sites, the site that owns the relocated object is ensured that the TE methods will be installed and executed at the new site exactly as they have been installed and executed at the previous site. The relocated object will therefore be as protected at its new site as it has been at its original site.

Despite the fact that an object will retain its original protection when it is relocated, a site may not always want any object to be relocated to any other site. In fact, the federal security policy is likely to allow each site to manage the relocation of objects

owned by the site. This means that a site S_1 may get a request to relocate (or replicate) an object owned by the site to another site S_2 (even though the object may currently be located at yet another site S_3). S_1 may decide to allow or prevent the relocation. If the relocation is allowed, it is possible for S_1 to change the protection of the object, or even change the object itself before relocating it. Using the example of the medical federated database above, a laboratory may only allow laboratory technicians to insert test data in the test report when the report resides at the laboratory. However, when the report is relocated to the hospital, the TE method associated with INSERTTESTDATA may be one that allows nobody to activate the method. Alternatively, this INSERTTESTDATA method may be removed from the object before the object is relocated; this, however, depends on the representation used for objects and also has some interesting implications for the inheritance and instantiation hierarchy used. It is a function of the TLE to give permission for the object to be relocated, to change the protection of the object (if required) and to adapt the object itself (if required).

Changing the object when it is relocated may mean that it may not be possible to reconstruct it later: If only methods are removed, the object can be reconstructed later from the class; if variables are removed from the object, it may be impossible to reconstruct the object later. (In the case of the hospital example above, it may be appropriate not to be able to totally reconstruct the test report object after it has been relocated to the hospital site.)

When replicating an object, the copy of the object (as well as its protection) may also be changed to reflect conditions at the replicated site. This introduces additional complexity that will be investigated in future research.

Snapshots may also be changed (and also be protected differently than the primary copy) before the snapshot is sent to its site. The inherent complexity is much less in this case than in the case of proper replication.

5 CAN SPO BE SECURE?

This section argues that SPO is not inherently too complex to be trusted.

As noted earlier, the TCC needs to be trusted by all members of the federation, while the TLEs and TEs only need to be trusted by their home sites. This implies the following:

1. The TCC needs to be small enough so that it can be verified easily enough;
2. No sensitive information should be passed to a TE or TLE (unless that information is owned by the home site of the TE or TLE);
3. The approach should not inherently require complex TLEs or TEs; and
4. The approach should not incur prohibitively high overhead costs, especially communicating between nodes.

The following paragraphs argue that requirements 1 and 2 can be met. Requirement 3 and 4 will receive attention later in the paper.

A prototype of an SPO system is currently being implemented (Olivier, 1995). This prototype focusses on the operation of the TCC and demonstrates that the TCC can be implemented using relatively short code: It consists of 27 services grouped into nine modules, with most of the services only a couple of lines of code long. Moreover, the

prototype has been designed such that one can argue about information flow and therefore determine what needs to be verified in each service, limiting the verification exercise. Therefore it does seem that requirement 1 above can be met.

The following remarks pertain to requirement 2. It is possible to build the TE such that it does not need access to any sensitive information at all: The TE is free to access data owned by its home site in any way allowed by the home site. However, if the TE wants to know, say, the current sensitivity level of the message all it can do is to request the TCC to abort execution if the message sensitivity is in a given range: after a TE method has executed, it may give the TCC a list of 'aborting conditions'—that is conditions that can be checked by the TCC and the message aborted if necessary.

An alternative solution to requirement 2 is followed by the prototype mentioned earlier (Olivier, 1995): TE methods are allowed access to sensitive information, but in an environment where such information cannot be stored or communicated to any other method in any way.

A TE can therefore be isolated from sensitive information, or restricted in what it can do with such information. However, it is sometimes necessary to send information to an TLE, and how the TLE uses the information cannot be restricted as easily. To illustrate, consider the case of LIBRARY above again. When a book's details are about to be changed, the relevant TE has to send a message to its TLE of the form *"Is the sender of the message a member of the* SUBJECTLIBRARIAN *group?"* In general, all such requests from the TE to its TLE will be of the form *"Does condition C apply to user U?"* or *"Does condition C apply to message M?"* Since the TE communicates with the TLE via the TCC, the TCC can easily ensure that all communication between a TE and TLE is limited to requests of the above format (and appropriate responses). Remember that such a request can only be sent to a TLE if the current message M, sent by user U, is attempting to access an object owned by the home site of the TLE. It would seem reasonable if a model provided a site with a list of users and messages that (attempt to) access objects owned by the site. There cannot be any criticism of SPO that only sends information to a site that allows this to be inferred.

Requirement 3 (regarding complexity) will be addressed in the next section. Requirement 4 (overhead costs) will be addressed in the conclusion.

6 TE METHODS

In section 3 the TE was identified as one of the components of SPO. The TE consists of a number of methods that are added to objects in the database to protect the objects. This section describes the use of such methods to protect entities in the database—in particular how these methods are associated with the entities they protect and how such TE methods can be reused to reduce the cost of security.

Secure object-oriented databases usually allow one to protect methods, instance variables, the object itself, or a combination of these (Olivier and Von Solms, 1994). SPO allows a TE method to be attached to any of these entities to protect the individual entity. (A site security policy may limit which of its entities may have TE methods associated.) The fact that SPO uses methods to protect entities means that a very general means of protection is available; however, it also means that the cost of security can be very high: Implementing secure methods can be expensive and executing a method for

every access can slow processing down considerably. The first concern will be addressed later in this section; the second concern will receive attention in the conclusion of the paper.

It is important to stress again that TE methods are not intended to be developed by the database programmer who implements the 'normal' database methods—the SSO will obtain them and associate them with the entities to be protected. We will briefly discuss this 'association' with entities, before the procedure to 'obtain' the TE methods will be discussed.

SPO does not prescribe a specific approach to be followed by a particular site for associating TE methods with entities—this may be done according the security policies of individual sites. However, one promising possibility would be for the SSO at that site to 'label' facets as they are introduced in the class hierarchy with the appropriate TE method (and 'add' the TE method to that class). To illustrate, assume that an organisation has an EMPLOYEE class with a GETSALARY method that may only be used by the personnel manager. The SSO can now add a TE method to this class to enforce this restriction (ABORTIFNOTPERSMANAGER) and associate this method with GETSALARY. Now the GETSALARY methods of all instances of EMPLOYEE can be protected by ABORTIFNOTPERSMANAGER. Further, if a subclass, PARTTIMEEMPLOYEE, of the EMPLOYEE class is defined, this new subclass may inherit GETSALARY, ABORTIFNOTPERSMANAGER and the fact that ABORTIFNOTPERSMANAGER protects GETSALARY from its superclass. Obviously, the method protecting an entity can be changed lower in a hierarchy—just like a database facet can be redefined—as long as the requirements for relationship restrictions given by Olivier and Von Solms (1994) are adhered to.

If the inheritance approach for TE methods described in the previous paragraph is followed, the same inheritance rules will apply to all facets of an object—whether they are normal database facets or security facets. However, it is suggested that the management of TE methods may not be handled by the compiler that changes normal database objects to their internal representation, but rather by a tool intended for this security function only; this tool will read the schema of the database, allow the SSO to label the objects and facets with the appropriate TE methods and produce output usable by SPO. The development of such a tool will not be addressed in the current work. Obviously such a tool can be used to enforce the relationship restrictions given by Olivier and Von Solms (1994).

As stated earlier, development of special TE methods can be costly. One possible approach is to note that most security restrictions will be of the form *"The current user can only activate this method (or access this variable) if the user is a member of group X"* where X can be specified; or *"The current message can only access this entity if it originated under circumstances C"* where C can again be specified and will typically concern the location from where the message was sent, the type of terminal from which the message was sent, the time the message was sent, etcetera. It should therefore be possible to have a very small library of generic TE methods such as ABORTIFUSERNOTMEMBEROFGROUP: G_1,G_2,G_3,...,G_n, or ABORTIFMESSAGE: CONDITION NOTONEOF: C_1,C_2,C_3,...,C_n. This would allow a site considerable flexibility when labelling its objects without the necessity to write any TE methods. Furthermore, if the use of a standardised library on a federation-wide base is mandated by the federal security policy, access checking can be optimised considerably—see section 8.

However, limiting TE methods to those from a library does limit the potential of SPO.

If TE methods are developed especially for specific classes, it is possible to implement content-based security: For example it may be possible to restrict access for GETSALARY of EMPLOYEE to the personnel clerk and personnel manager if the salary is \$10 000 or less, but only to the personnel manager if the salary exceeds \$10 000. Since the TE is a method, associated with the protected object, it has access to the variables of the object and therefore can perform such content-based checks.

In this paper we have assumed that the sites of the federation are homogeneous. However, if sites use dissimilar hardware or operating software the transfer of TEs between sites may cause some concern. This may be dealt with in a number of ways. Firstly, if TEs are all of some standard form as described above, it is not necessary to transfer executable code between sites—it is only necessary to send an indication that, say, the TE to be executed is of the form ABORTIFUSERNOTMEMBEROFGROUP: G_1,G_2,G_3,...,G_n and supply the parameters G_1,G_2,G_3,...,G_n. The prototype described by Olivier (1995) uses another approach: The TCC includes a small interpreter for a simple postfix notation. All TEs are expressed in this notation and can therefore be interpreted by all sites independent of their physical characteristics.

7 FORMALISATION

Formalisation of the concepts used in SPO are required. This section describes an approach to such a formalisation.

A vector of attributes $\overline{\alpha} \in A_1 \times A_2 \times A_3 \times \ldots \times A_n$, as required by the federal security policy, is associated with each message. Here, A_1 may, for example be the identification of the user who initiated the current exchange of messages, A_2 the clearance of the message, A_3 the current sensitivity of the message, and so on. Additionally, a vector of attributes $\overline{\beta_i} \in N \times B_{i,1} \times B_{i,2} \times B_{i,3} \times \ldots \times B_{i,m}$ may be associated with messages that originate from a site i. The first entry of this vector identifies the site from which the message originated; later entries may include information such as the current role of the user who sent the message and the identification of the terminal from which the message was sent. The exact composition of each $\overline{\beta_i}$ depends on the security policy of site i. The pair $(\overline{\alpha}, \overline{\beta_i})$ accompanies each message sent, either logically, or physically. Note that the TLE at the originating site i will provide the initial values for both ($\overline{\alpha}$ and $\overline{\beta_i}$). However, after message sending has been initiated, only the TCC is allowed to modify $\overline{\alpha}$. Only the TLE at site i (or the TCC, on behalf of TLE_i) is allowed to modify $\overline{\beta_i}$.

To perform access control, the TCC compares each attribute a_j in $\overline{\alpha} = (a_1, a_2, a_3, \ldots, a_n)$ with the set of valid values for the entity to be accessed. Similarly, each attribute $b_{i,j}$ in $\overline{\beta_i} = (i, b_{i,1}, b_{i,2}, b_{i,3}, \ldots, b_{i,m})$ is compared with the valid values for the entity to be accessed as specified by the TE associated with that entity. If all the attributes are within the specified ranges, access is allowed. The federal policy may also specify how $\overline{\alpha}$ is to be modified because of the access, while the policy of site i may specify how $\overline{\beta_i}$ is to be modified prior to the access. More formally, the federal security policy specifies a vector of conditions (or valid values) $\overline{A} = (A'_1, A'_2, A'_3, \ldots, A'_n)$ with each $A'_j \subseteq A_j$. Similarly, the site security policy at every site i specifies a vector of valid values $\overline{B_i} = (B'_{i,1}, B'_{i,2}, B'_{i,3}, \ldots, B'_{i,m})$ with each $B'_{i,j} \subseteq B_{i,j}$. If a request from site i arrives at any site and attempts to access an entity protected by a pair $(\overline{A}, \overline{B_i})$ access will only be granted if for every a_j in $\overline{\alpha}$, $a_j \in A_j$, and,

for every $b_{i,j}$ in $\overline{\beta_i}$, $a_{i,j} \in A_{i,j}$. However, if the entity to be accessed is protected by the pair $(\overline{A}, \overline{B_k})$, with $i \neq k$, $\overline{\beta_i}$ cannot be compared directly with $\overline{B_k}$. This will be considered later in this section.

In the preceding paragraphs, the assumption has been that the pair $(\overline{\alpha}, \overline{\beta_i})$ accompanies the concerned message. However, in practice, many of the attributes may not always accompany the message, but only be requested when required. For example, the terminal type may be requested from the concerned TLE if it is required for a specific access check. This has been the assumption earlier, in, for example, figure 2 where additional information was requested from the owning TLE before access could be granted.

To illustrate access checking further, assume that sites LIBRARY and LIBRARY2 both have subject librarian groups. Now assume that user U at LIBRARY attempts to modify an object BOOK2 owned by LIBRARY2. The TCC will check the federal access constraints. After this, the attributes in $\overline{\beta}_{Library}$ that 'accompanied' the message will be compared to the TE method protecting BOOK2. However, since BOOK2 is owned by LIBRARY2, it would be necessary to compare $\overline{\beta}_{Library}$ to restrictions of the form $\overline{B}_{Library2}$. Here, the required information about the message will be requested from the TLE at LIBRARY2. If LIBRARY2 has a bilateral agreement with LIBRARY, it will request this information from LIBRARY, format it appropriately, and present it to the TCC to complete the access checks.

Space restrictions prevent us from considering such mappings between the vector formats of different sites in more detail. This, as well as other aspects such as formalisation of content based access checks and formalisation of cases where a user from one site accesses the database from another site (where group and other information about the user may not be available) will be addressed in future work.

8 CONCLUSION

The paper described an implementation strategy for self-protecting objects in a secure federated database. Since it is an implementation strategy it is intended to be usable with more than one model (and therefore more than one set of security policies).

The overhead costs of a secure federation using SPO can be high. However, it is possible to reduce overhead costs—in some cases trading generality for speed:

- It was stated that TE methods may accompany objects when they are relocated. Although this is not absolutely necessary, it is recommended since it will reduce (or eliminate) the sending of messages between sites during an access check.
- The federal security policy can restrict the available TE methods to only a few predefined ones as described in the text. In such a case access checking can be as efficient as in the access control methods currently used (usually access control lists and capabilities).
- Even if the TE methods are not restricted in any way, the fact that the same TE method will probably be inherited from the point in the class hierarchy where it is defined down to all instances using it, means that such code will be reused—adding to the possibility to verify and optimise it and, hence, to increase the level of trust in them.
- If an access check has been performed, it may not be necessary to perform an identical check again, but just use the previous decision again. Such 'caching' of checks may

increase the speed of access checking considerably, but may limit the possible access checks that may be used and/or compromise security. Content-based checks may, for example, cause problems if caching is used.

Note that it is also possible to add a number of other federally trusted services to SPO—as an example it may prove useful to add a federally trusted authentication service (possibly based on Kerberos; see Coulouris, Dollimore and Kindberg, 1994:495–502 for an introduction). Obviously a federally trusted communications facility is required to link the various TCCs and a federally trusted authorisation service may be used. These components will be investigated in future.

It is also interesting to point out that the prototype (Olivier, 1995) uses a multilevel federal security policy (TCC) that may be augmented by multilevel and discretionary local security policies. However, implementation of multilevel site security policies on a TCC that only provides discretionary security seems to be problematic. More work needs to be done here.

We have also assumed that objects do (physically) encapsulate their methods. In practice, the code for methods are not duplicated for every object, but rather reused from the class definition. We have also pointed out that TE methods may be reused in many cases. In both cases it may not be necessary to copy all methods to a new site when an object is relocated. This may be done using a mechanism similar to that described by Lee, Wang and Chang (1994). The required changes to their approach in a secure environment will be considered at a later stage.

A number of other aspects that deserve further research attention have been identified in the text. Those that we are currently devoting our attention to include further work on the prototype (Olivier, 1995) and investigation of distributed authorisation.

REFERENCES

Bull, JA, Gong, L and Sollins, KR (1992) Towards Security in an Open Systems Foundation, 3–20 in *Computer Security—ESORICS 92. Second European Symposium on Research in Computer Security*, (eds Y Deswarte, G Eizenberg and J-J Quisquater) Springer-Verlag, Amsterdam.

Castano, S, Fugini, MG, Martella, G and Samarati, P (1995) *Database Security*, Addison-Wesley, Wokingham, England.

Ceri, S and Pelagatti, G (1985) *Distributed Databases*, McGraw-Hill, NewYork.

Coulouris, G, Dollimore, J and Kindberg, T (1994) *Distributed Systems: Concepts and Design, Second edition*, Addison-Wesley, Wokingham, England.

Date, CJ (1985) *An Introduction to Database Systems Volume 2*, Addison-Wesley, Reading, Massachsetts.

Denning, DE (1988) Database Security, 1–22 in *Annual Review of Computer Science Volume 3* (eds JF Traub *et al*), Annual Reviews Inc, Palo Alto, California.

Gong, L and Qian, X (1994) The complexity and Composability of Secure Interoperation, *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, 190–200, Oakland, California.

Idris, NB, Gray, WA and Qutaishat, MA (1994) Integration of Secrecy Features in a Federated Database Environment, 89–108 in *Database Security VII: Status and Prospects*,

(eds TF Keefe and CE Landwehr) North-Holland, Amsterdam.
Jajodia, S, McCollum, CD and Blaustein, BT (1994) Integrating Concurrency Control and Commit Algorithms in Distributed Secure Databases, 109–21 in *Database Security VII: Status and Prospects*, (eds TF Keefe and CE Landwehr) North-Holland, Amsterdam.
Jonscher, D and Dittrich, KR (1994) An Approach For Building Secure Database Federations, *20th VLDB Conference*, Santiago, Chile.
Kim, W (1991) Object-oriented database systems: strengths and weaknesses, *Journal of Object-oriented Programming*, **4**, 4, 21–9.
Lee, YS, Wang, FJ and Chang, CH (1994) Object Movement in Distributed Object-oriented Systems, *Information Sciences*, **78**, 19–47.
Olivier, MS and Von Solms, SH (1992) Building a Secure Database Using Self-protecting Objects, *Computers & Security*, **11**, 3, 1992, 259–71.
Olivier, MS and Von Solms, SH (1994) A Taxonomy for Secure Object-oriented Databases, *ACM Transactions on Database Systems*, **19**, 1, 3–46.
Olivier, MS (1995) A Multilevel Secure Federated Database, *Database Security VIII (A-60)*, 183–98, (eds J Biskup, M Morgenstern and CE Landwehr), North-Holland, Amsterdam.
Olivier, MS (1995) Self-Protecting Objects: A Prototype, *Submitted.*
Özsu, MT and Valduriez, P (1991) *Principles of Distributed Database Systems*, Prentice-Hall, London.
Pernul, G (1993) Canonical Security Modelling for Federated Databases, 207–22, in *Interoperable Database Systems* (eds DK Hsiao, EJ Neuhold and R Sacks-Davis), Elsevier, Amsterdam.
Rabitti, F, Bertino, E, Kim, W and Woelk, D (1991) A Model of Authorization for Next-Generation Database Systems, *ACM Transactions on Database Systems*, **16**, 1, 88–131.
Sheth, AP and Larson, JA (1990) Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, *ACM Computing Surveys*, **22**, 3, 183–236.
Thuraisingham, B (1994) Security issues for federated database systems, *Computers & Security*, **13**, 509–25.
Wegner, P (1990) Concepts and Paradigms of Object-Oriented Programming, *OOPS Messenger*, **1**, 1, 7–87.

9 BIOGRAPHY

Martin Olivier holds a Ph.D. in Computer Science and is currently a senior lecturer in Computer Science at the Rand Afrikaans University in Johannesburg, South Africa. Current research interests include database security, especially for object-oriented and distributed databases.